School of Computer Science

Final Year Project

# Formal Report

Submitted in fulfilment of the requirements for the BSc (Hons) Degree in

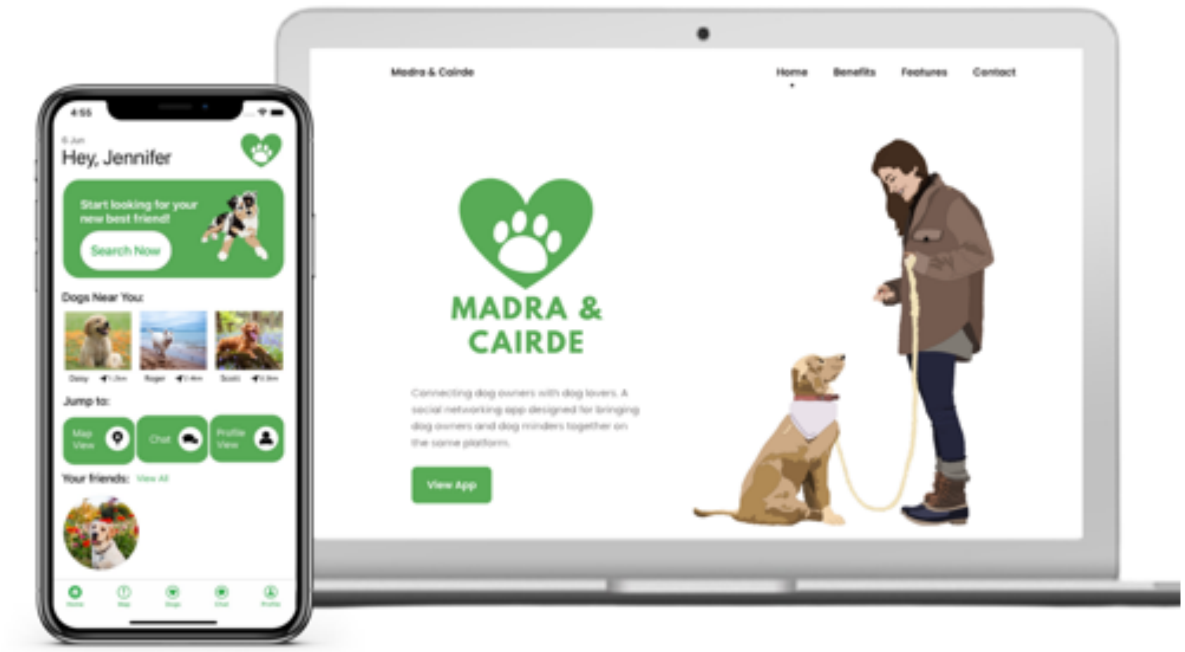
Computer Science & Information Technology

[redacted]

# Madra agus Cairde App

# MADRA & CAIRDE

[redacted]



Connecting dog owners with dog lovers. A social networking app designed for bringing dog owners and dog minders together on the same platform.

# Table of Contents

## Acknowledgements

I would like to express my gratitude to my supervisor, Karen Young, who helped me complete this Final Year Project. Her continuous support throughout final year was crucial for the completion of this project. I would also like to thank my user testers who helped me at each iteration of the process with their feedback. Lastly, I would like to thank my family and friends who encouraged me to give it my all. They also participated in numerous user tests for me, which contributed enormously to the overall project

## Glossary

| | |
|---|---|
| FYP | Final Year Project |
| UX | User Experience |
| UI | User Interface |
| CI | Continuous Integration |
| CD | Continuous Development |
| MVC | Model-View-Controller |

# Chapter 1. **Introduction**

## 1.1. **Problem Statement**

This Final Year Report provides the context, description and documentation of the mobile application and website developed for the Final Year Project (FYP) as part of the Computer Science and Information Technology degree. This report will cover the entire lifecycle of the project – from start to finish. It will provide information and details on the background research involved; the design elements used as well as the functional requirements of the application. The skills that were required to manage such a big undertaking will be discussed. Any implementation details will be analysed in a comprehensive manner so the reader will gain a full understanding of the project. To draw the report to a close, an overall perspective of the FYP completed will be given and any downfalls or future plans for the project will be reviewed.

The title of this FYP is 'Madra & Cairde'. The dominant part of this project is a mobile application for iOS. A website also was created to support and promote the 'Madra & Cairde' application. The main goal of Madra & Cairde is to allow people to look for dogs in their locale and organise a play date with them.

The project will address the need of a significant cohort of Dog Owners and Dog Minders, and will provide an interface appropriately designed for each of the two categories. A mobile app will be iteratively designed with the help of user feedback, to ensure the best possible product is produced.

## 1.2. **Background**

This project idea came about after lots of thinking and research. A niche mobile application idea was sought as it was felt that a new and interesting idea would provide motivation for the effort required. The process began by identifying interests, identifying related problems that could be solved, and looking at similar applications/businesses in the public realm.

One big area of interest and passion is animals. Experience shows that some people have never had a pet but love going to family and friends' houses where pets live. Social media feeds are filled with animals; from Instagram's explore page to TikTok's video queue, much of which concerns one type of animal in particular – dogs.

One problem easily identified from experience is missing sufficient interaction with dogs. This struggle was confirmed by friends, even those with pets, because being away from home at college means only rare visits to pets.

From this sprang the idea that it would be good for people to be able to borrow dogs for play dates. It would be a good service for both the borrower and the dog owner for a number of reasons. Lots of owners do not have the time to walk their dog or keep it company while they're working.

The main target audience for the dog borrower user (i.e. Dog Minder) is college students as they are likely to avail of such a service and would benefit greatly from it. College students these days are very tech-savvy and are well versed in social media and social networking apps. This means that in offering such an intriguing service, the app could be easily adopted by them as they are already used to using an abundance of apps.

A high number of college students suffer from mental health issues. According to a report done by The Union of Students in Ireland, ‘Students are experiencing extremely severe levels of anxiety (38.4%), depression (29.9%) and stress (17.3%)’. There is a great number of students who do not have a proper outlet to discuss or vent about these personal difficulties. [1]

It is scientifically proven that exposure to a dog, or ‘puppy love’ as many people refer to it, can help reduce stress and anxiety as well as many other things. The love and warmth from a dog can ‘buffer depressive tendencies, quell anxious feelings, and mitigate loneliness’.[2] Walking a dog is a great way to encourage exercise, especially if the person is unlikely to go for a walk for any other reason. Exercise is another proven way of improving one's mental health and mood. [3]

The main target audience for the Dog Owner user would be adults/couples with a pet dog who may not have the time to walk their dog or are anxious about leaving their dog home alone while they go somewhere/go to work.

There has been a huge increase in separation anxiety in pets, especially since the beginning of the pandemic which has had most people working from home. Most pets have now become used to and reliant on their owner being with them all of the time. Because of this, it will be extra tough on pets when their owner has to return to their workplace and leave them unaccompanied for hours on end. This will inevitably cause a lot of stress and anxiety in most pets. [3] As with humans, pets benefit a lot from daily exercise. A healthy dog is a happy dog. [4] Unfortunately, some Dog Owners do not have enough time to walk their dog; this may be due to work commitments or other life stresses. [5]

The *Madra & Cairde* app will facilitate both the mental and physical well-being of the Dog Lover and the dog, as well as providing a helpful service for Dog Owners so they can have peace of mind to know that their dog is not lonely while they are busy.

## 1.3. **Stakeholders**

The sole stakeholders in this app are the end users. Madra agus Cairde’s end users are separated into two categories, the **Dog Owner** and the **Dog Minder**. Other types of stakeholders could conceivably be catered for if the project were to be developed further.

However, as of now, all the work that has been put into the project has been done with these end users in mind. The entire project has been consistently iterated over the course of the development process to fit better with results of user tests and user feedback. After all, without end user feedback being incorporated into the design, the author would not have identified the full list of requirements the app needed.

## 1.4. **Project Management**

Good project management is essential to ensure that required features are delivered in a timely manner. The project needs to be planned in outline to get an estimate of the effort required, then tracked against the resultant timeline to make adjustments to the plan. If tasks proceed more quickly than expected, some of the additional non-core tasks can be tackled. Conversely, if tasks take longer than expected, they can be revised in scope or relegated to non-core requirements.

Despite being a single-developer project, the Agile development process was originally applied. This entailed grouping tasks into a series of "sprints" (time periods of a fixed duration; usually a couple of weeks). However, the tasks proved unpredictable in duration because of the learning required for many of them, and commitments to other parts of the degree course, like assignments.

Accordingly, the project evolved into a test-driven development cycle, where each task was worked on until it passed a set of tests, then the next task was started. This still required a periodic assessment of the project's progress, with refinements of the goals a frequent occurrence.

## Trello

Trello was chosen to keep track of tasks at various stages. It was used to group tasks into:

- those yet to done
- those in review
- those that were completed
- bug triage
- bugs that were fixed
- extras
- not doing

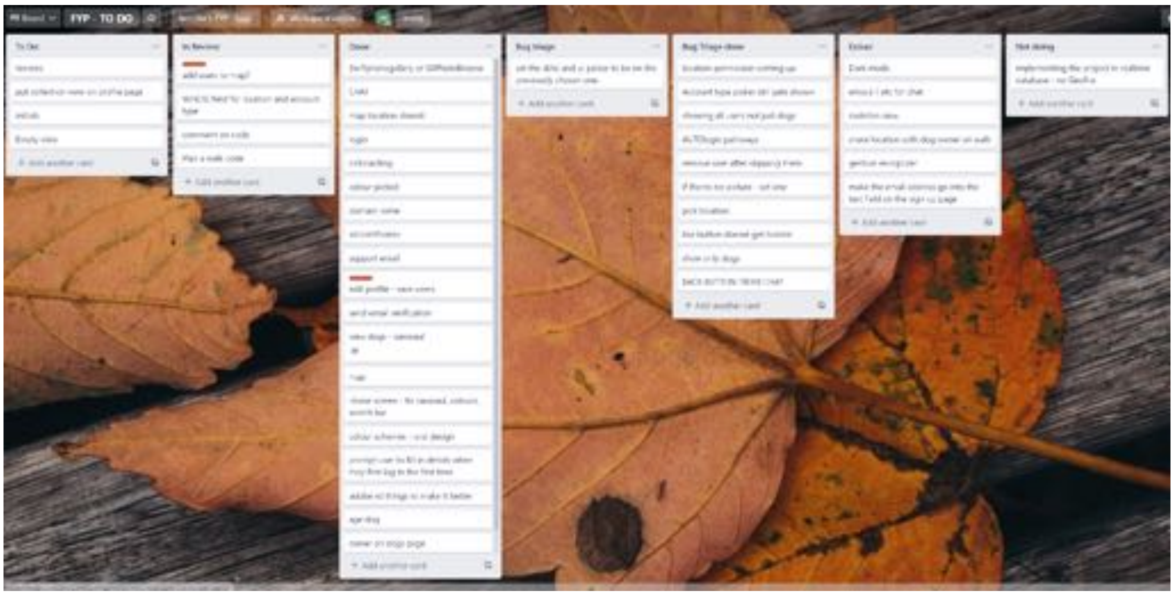


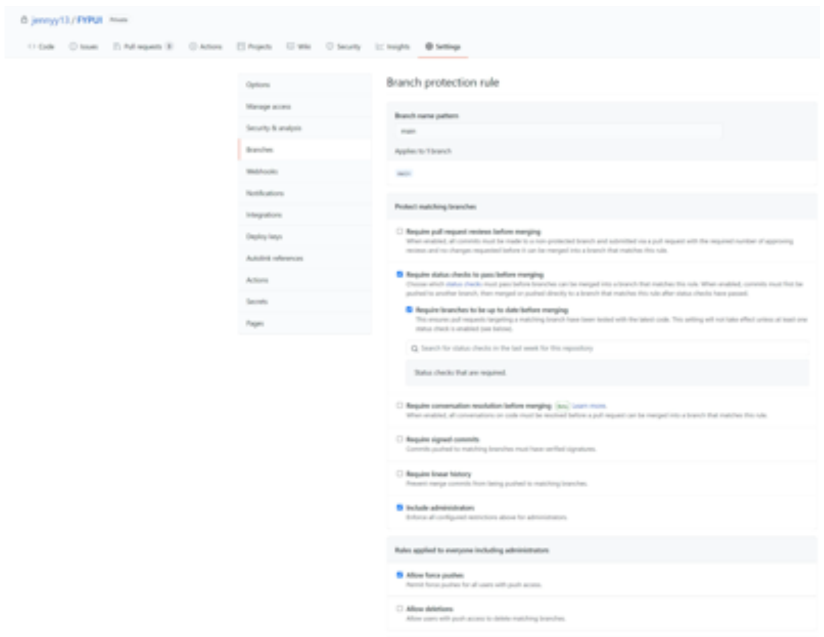
*Figure 1: Trello Board*

Trello was chosen because of familiarity with it from having used it during professional placement, where different boards were used to keep track of work done in different sprints and tech debt. Trello ensured that particular tasks were not forgotten about so the size of the project was always apparent. It proved very useful in getting back onto the project when breaks needed to be taken to complete other college work.

## Git

Git was used for version control, commits and pull requests (PR). A private repository was created in GitHub to connect to the local git repository on a working computer. Git and GitHub were used while on professional placement so the author had grown accustomed and familiar with them. Git is a very good tool which can be used seamlessly to keep track of any iterations done on blocks of code. If any big mistakes are made, it is possible to revert to a commit where the code is not broken.

Another way that good project management was sustained through the project was by using clear branching strategies. Branching is a very important concept in Git. The very core requirement necessary for the Github flow to work correctly is that the main branch should always be working and deployable. This means that code on main should never be broken as this would stunt future work. To ensure this was always the case, a set of branch rules was devised. This required all new PRs to pass the unit tests that were put in place, before the option to merge would appear. This stopped poor/broken code from making its way into main.



Branch protection rule

Branch name pattern: main

Applies to 1 branch: main

Protect matching branches

- [ ] Require pull request reviews before merging
  [illegible]
- [x] Require status checks to pass before merging
  [illegible]
  - [x] Require branches to be up to date before merging
    [illegible]
  - Search for status checks in the last week for this repository
  - Status checks that are required.
- [ ] Require conversation resolution before merging
  [illegible]
- [ ] Require signed commits
  [illegible]
- [ ] Require linear history
  [illegible]
- [x] Include administrators
  [illegible]

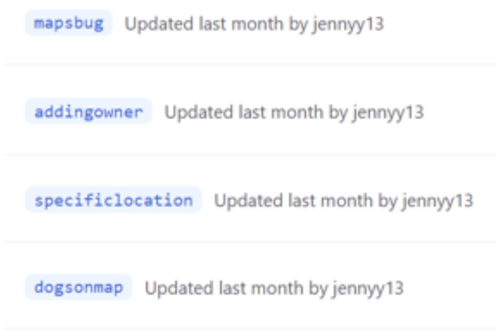
Rules applied to everyone including administrators

- [x] Allow force pushes
  [illegible]
- [ ] Allow deletions
  [illegible]

*Figure 2: Git Branch Rules*

New branches were always derived from main to start off as base code, and clear and descriptive titles were given so the author was clear about what concept/feature was being worked on.

Examples of these titles which made it clear to the author what the branch should do, are:



mapsbug Updated last month by jennyy13

addingowner Updated last month by jennyy13

specificlocation Updated last month by jennyy13

dogsonmap Updated last month by jennyy13

*Figure 3: Branch names*

In Git, when making a commit, there is an option '*-m*' to add a comment to the commit. This was another good management practice used throughout the development of the application as it meant the author could tell from a PR what exact work was done in that PR.

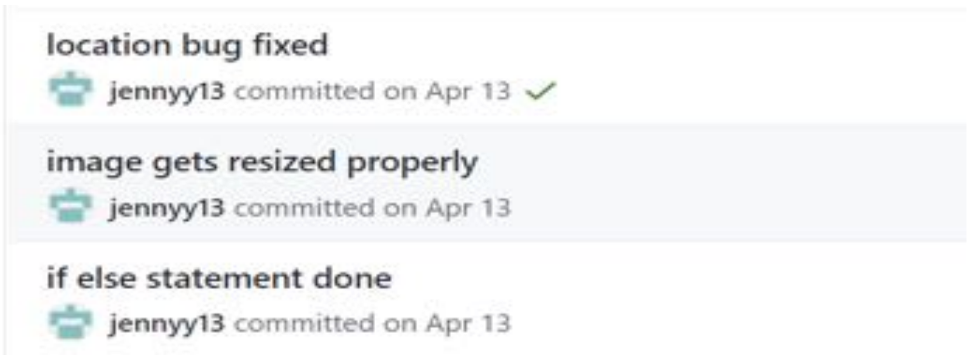

location bug fixed
jennyy13 committed on Apr 13 ✓

image gets resized properly
jennyy13 committed on Apr 13

if else statement done
jennyy13 committed on Apr 13

*Figure 4: Commit captions*

Another strategy that was used by the author was the use of a template for PRs. The format of the template was simple yet effective:

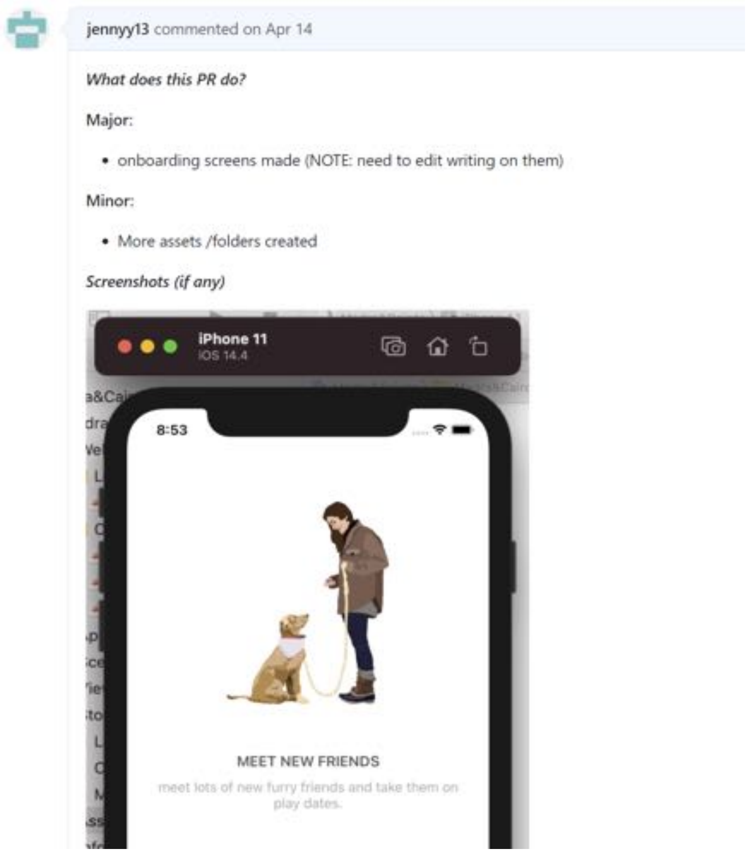
*What does this PR do?*

*Major:*

*Minor:*

*Screenshots (if any)*

An example of this in play is:


jennyy13 commented on Apr 14

*What does this PR do?*

Major:

- onboarding screens made (NOTE: need to edit writing on them)

Minor:

- More assets /folders created

*Screenshots (if any)*

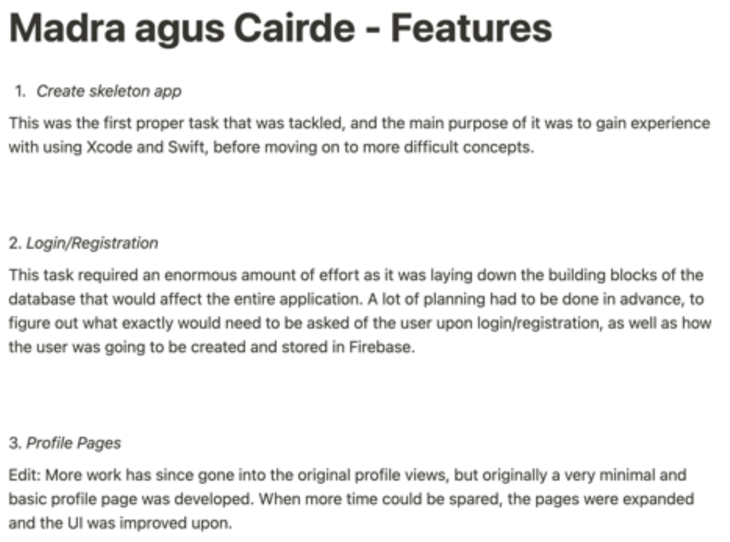
*Figure 5: PR Template*

Although there was only one developer working on this project, it was still a good practice to have in place. This was a large project with many different components which meant it would be easy to lose track of work completed if good project management was not in place.

**Plan**

A Gantt chart was originally made to be followed and was included in the project definition document. However, as more Swift and iOS development experience was gained, the original timeline had to change. The original ordering of the tasks was used (more or less) as it made the most logical sense to develop the application that way. A big aspect of starting out in a new development environment is that things cannot be estimated easily. Features that were expected to take a few hours could end up dragging on for weeks because inexperience made it harder to deal with errors and bugs, and just in general, having to learn new things makes the development process a lot longer. Also, the original Gantt chart did not account for challenges that could arise, such as technical difficulties and database complications etc.

As will be discussed in Chapter 3, a Test-Driven Development approach was used when developing this project. In simple terms, this meant that features were focussed on for as long as was necessary to make them work, and then the next feature was tackled. Once each main feature was completed, notes were kept to record the work that was completed and lessons that were learned. To record these notes in a safe and secure environment (i.e., where it won't accidentally be lost), an account was created with Notion.



**Madra agus Cairde - Features**

1. *Create skeleton app*

This was the first proper task that was tackled, and the main purpose of it was to gain experience with using Xcode and Swift, before moving on to more difficult concepts.

2. *Login/Registration*

This task required an enormous amount of effort as it was laying down the building blocks of the database that would affect the entire application. A lot of planning had to be done in advance, to figure out what exactly would need to be asked of the user upon login/registration, as well as how the user was going to be created and stored in Firebase.

3. *Profile Pages*

Edit: More work has since gone into the original profile views, but originally a very minimal and basic profile page was developed. When more time could be spared, the pages were expanded and the UI was improved upon.

4. *Dog Display*

Implemented in the application are two ways in which the dogs are displayed. A Udemy tutorial for a Tinder style app (here: https://www.udemy.com/course/ios14-tinder-like-dating-application-with-firebase-swift/ ) was followed originally to implement the card view (swiping view). This was a crucial tutorial for gaining an understanding of how to download and interact with users from the database. This information and newly gained understanding were able to be used to develop another type of way to display dogs (i.e., the county search).

5. *Friend Match*

Once the search functions were implemented, the ability to send and accept friend requests was implemented. This step needed to be done prior to the messaging service being implemented, as a key feature of this app is security which means that only friends can message each other. This is in an effort to protect users from unwanted messages if they are not interested in speaking to a fellow user. Implementing the friend requests took longer than expected as there were a lot of minor details that needed to be implemented alongside. For example, once users become friends, the UI needs to be updated so that the 'add friend' button is removed and, in its place, a chat button appears. This task was not originally included in the Gantt chart, as the need only became clear as a deeper understanding of the app evolved.

6. *Messaging Service*

The messaging service was expected to take a long time as it is clearly a big and data-complicated task. However, unfortunately, still not enough time was allocated in the original plan to fully deal with this. The messaging service allows friends to communicate with each other and send media links (e.g., pictures) to one another.

7. *Map*

The map feature caused a serious problem. It was during this stage of development that it was discovered that the library 'GeoFire' which would accomplish the feature of displaying nearby users, was incompatible with the database that had been used to build the entire project up until this point. A lot of investigation into other alternatives was done, but they all led to dead ends. It was a matter between choosing to leave that feature and not implementing it, or to restart the entire project using the other kind of database that supports 'GeoFire'. A new project was briefly made but it was soon realised that translating all of the work done on the original project was pretty much impossible and the app would need to be started from scratch if development was going to continue successfully. Everything was weighed up and it was decided for the good of the project, that feature would be left out, and work on other features would continue on the original branch. It would have been too complicated to learn an entirely new way of coding and everything learned up until then would have gone to waste as there are enormous differences between the way Firestore Database is used compared to Realtime Database.

8. *Push Notifications*

Research and development was done, but unfortunately it could not be fully implemented. Error: 'Provisioning profile does not support the push notifications capability'. There is membership of the iOS University Developer Program and team is set to the NUIG team and it still will not work. Not enough time to fully investigate this issue, so push notifications will not be implemented.

9. *Testing*

Testing, both user and system, was implemented all throughout the development process, however it was ramped up particularly towards the end of development. Results from this testing contributed to any extra iterations on the code as well as on how the UI/UX was changed before completion.

# Chapter 2. **Requirements**

## 2.1. **Introduction**

The project as originally envisaged proved overly ambitious for the amount of learning required, not to mention technical difficulties which arose, some of which were insurmountable. Therefore, the list of core requirements was gradually pruned. This chapter describes which tasks remained in the core list, and those which were relegated to a secondary, non-core, list.

## 2.2. **Functional Requirements**

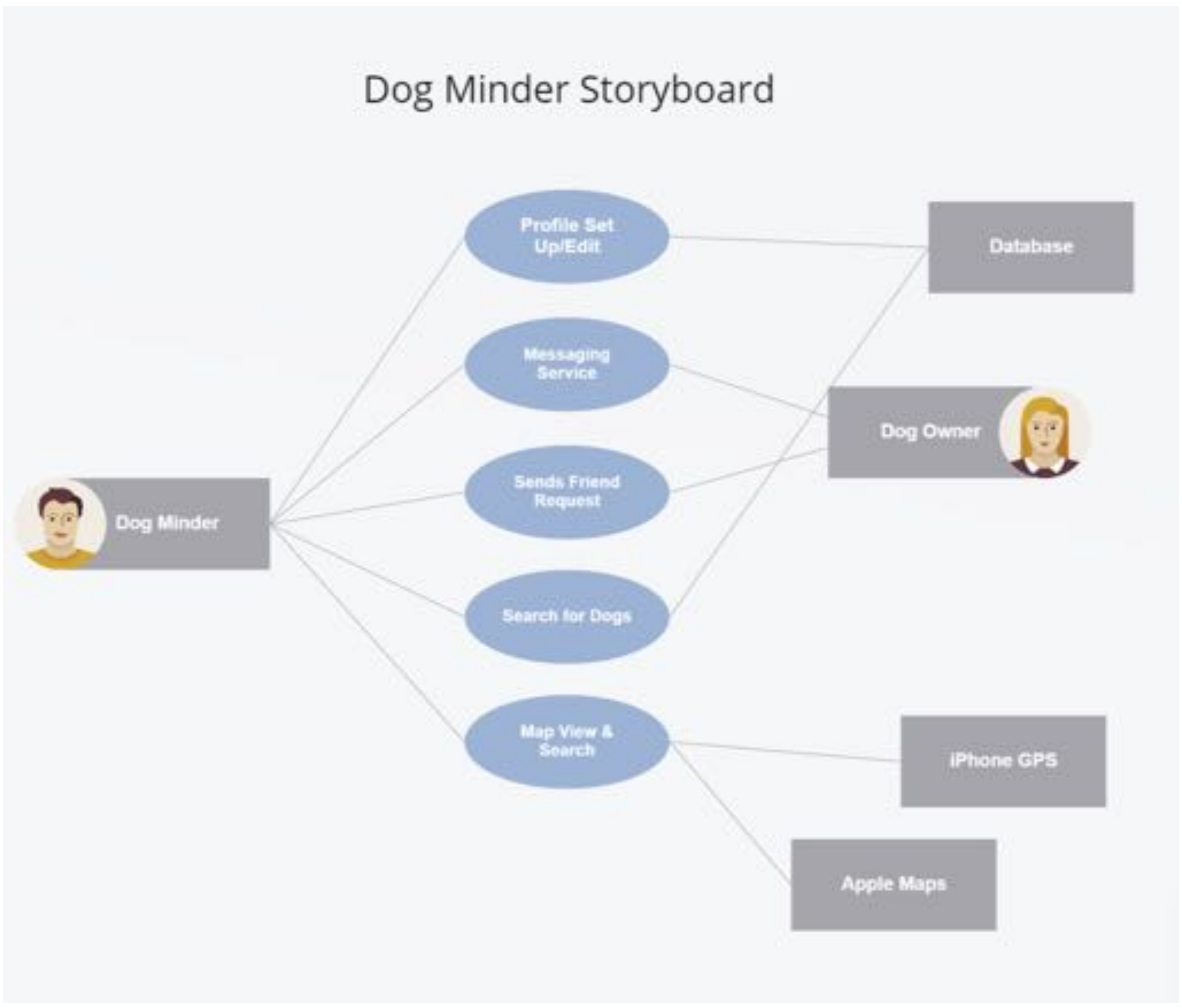


*Figure 6: Dog Minder Storyboard*

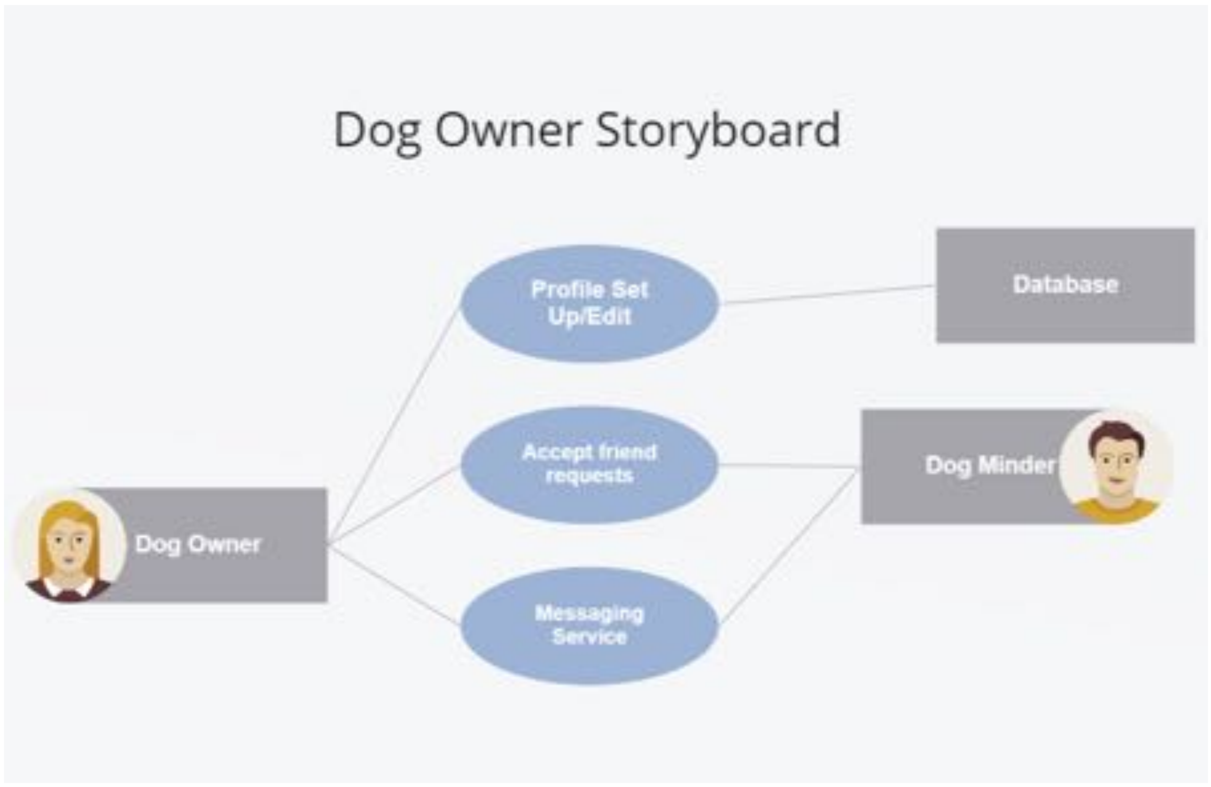


*Figure 7: Dog Owner Storyboard*

The core requirements of the app are:

- Profile Set-up
- Messaging Service
- Add Friends
- Accept Friends
- Search

The non-core requirements are not essential to the basic functionality of the app but they do add features which contribute a lot to the overall appeal of the app. Those are:

- Onboarding Screens
- Map View (Location Sharing & Search)
- Card View (i.e., Swiping)
- Photo Upload (Select from Gallery & Camera Use)
- AutoLogin

### Profile Set-Up

This was a core requirement as a user must be able to set up a profile in order to use the app. There is a process for initial set up when a user account is first created. Extra information can be added via the Edit Profile page, which is accessed through the edit button on the profile page. Information that is expected to change, such as location and ‘about’ can be edited. However, information supplied at the beginning, such as name and breed etc, cannot be modified.

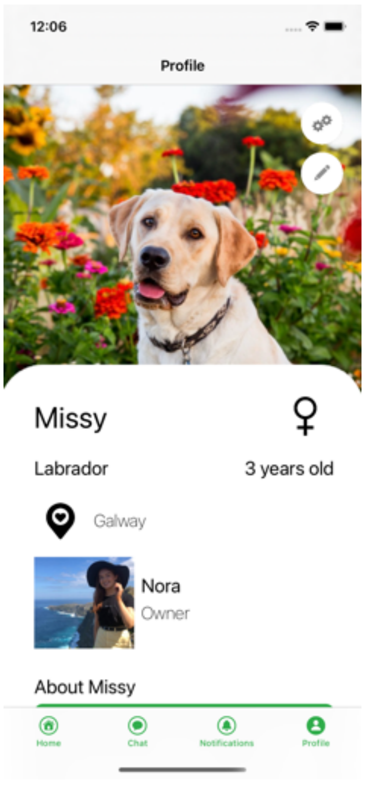


*Figure 8: Example of Dog Owner Profile*

## Messaging Service

Another core requirement necessary in the app for full function is the messaging service. Users need to have the ability to communicate with each other – that is the main purpose of the app.

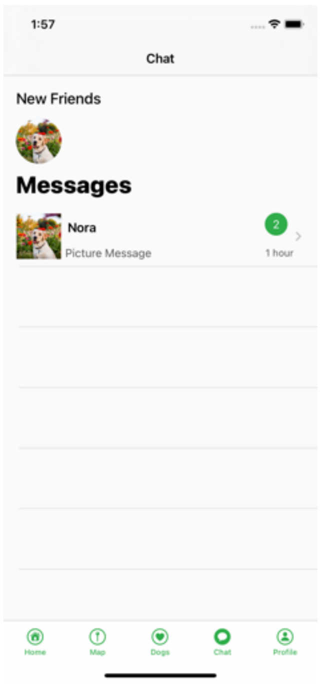
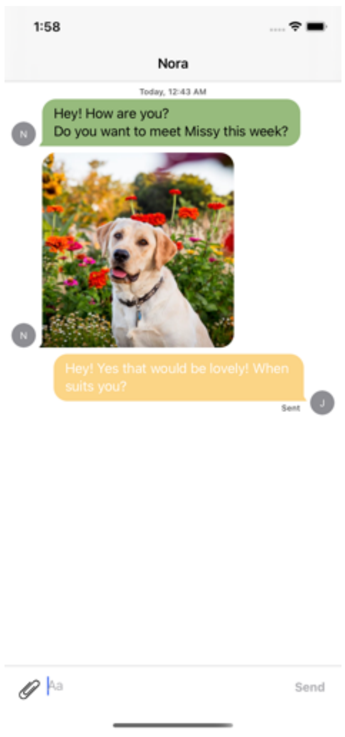


*Figure 9: Example of messaging*



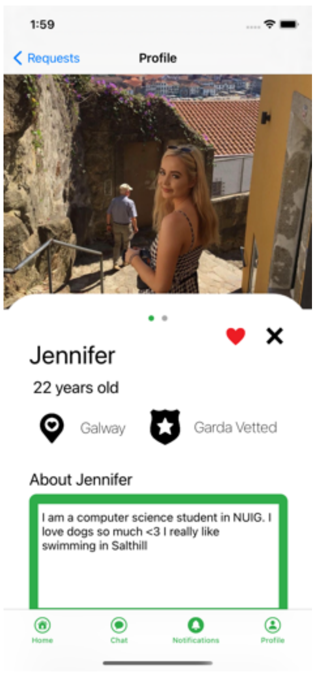
Hey! How are you?
Do you want to meet Missy this week?

Hey! Yes that would be lovely! When suits you?

*Figure 10: Messages between users (including picture message)*

## Add Friends

Conversations can only occur between users that are friends, so in order to become friends, a Dog Minder must be able to send a friend request to the Dog Owner. There are two ways of adding friends in the app. The main way is by clicking the heart button that appears on user profiles. The alternative is by swiping the user card left if searching is taking place via the card deck (accessed via the random search button).



Jennifer
22 years old
Galway
Garda Vetted
About Jennifer
I am a computer science student in NUIG. I love dogs so much <3 I really like swimming in Salthill

*Figure 11: Example of profile with accept/reject buttons*

## Accept Friends

Once a friend request from a Dog Minder has been sent to the Dog Owner, the owner needs to have the ability to accept (or reject) this request. The Dog Owner may accept the message by similarly pressing the heart button on a user's profile. Once a Dog Owner does this, the friendship is formed and both users appear on one another's home screens/chat list.

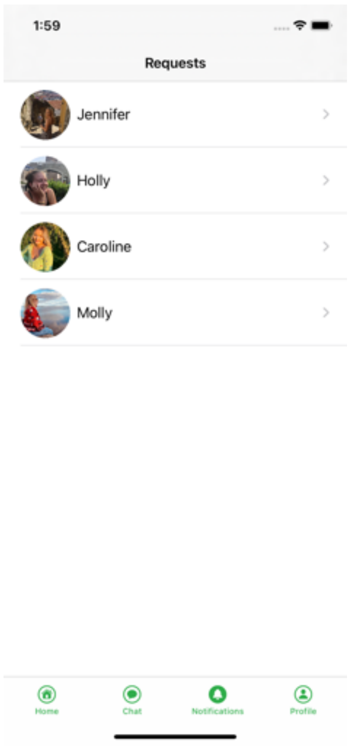


*Figure 12: Friend requests*

## Search

Search an essential element of this app. The user must be able to search through users in the database, and filtering by location if necessary. The search may done via the random search, as previously mentioned, or the location search.

These are the requirements necessary for the app to function at a basic level. In order to improve the user experience and overall quality of the app, another of other features were implemented.

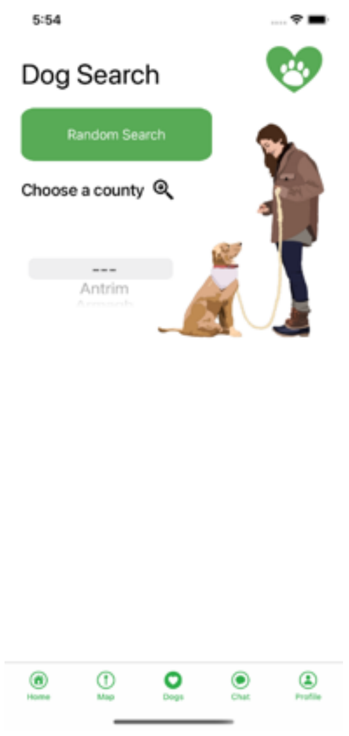


*Figure 13: Dog Search View*

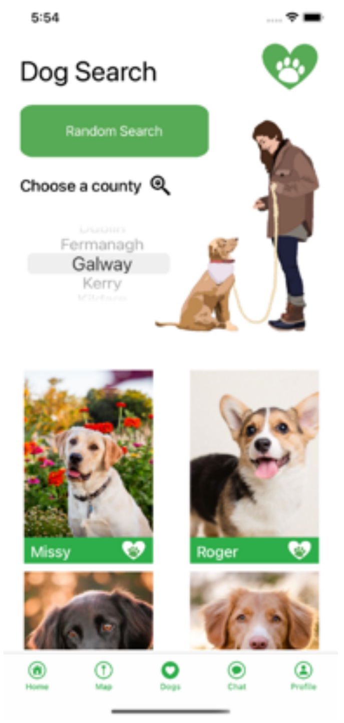
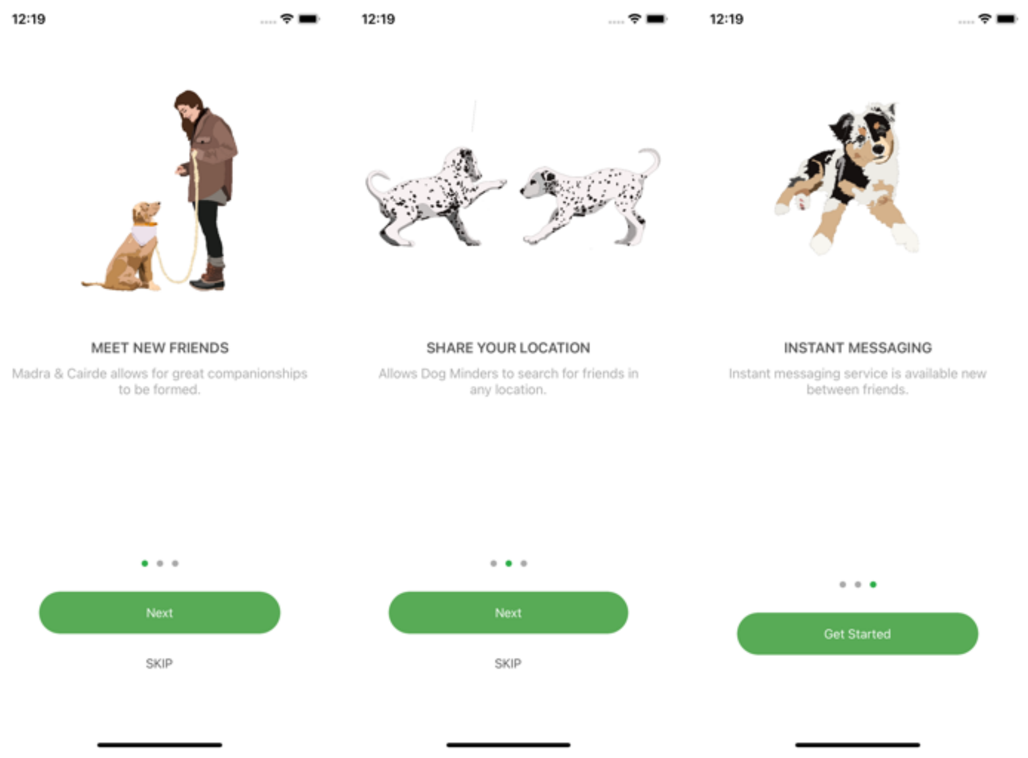


*Figure 14: Dog Search View after county search for 'Galway'*

Onboarding

Onboarding is important for a user to get a sense about what is involved in the app. It is a chance to educate the users about the features they will get to use in the app.

A key to ensuring a good onboarding experience is to keep the messaging clear and concise. It is also important that a user has a chance to opt out of the onboarding, i.e., to skip past the onboarding screens if they wish.



Madra & Cairde allows for great companionships to be formed.

Allows Dog Minders to search for friends in any location.

Instant messaging service is available new between friends.

*Figure 15, 16, 17: Onboarding Screens that appear when app is first opened*

The onboarding screens in Madra agus Cairde help give users a glimpse as to what the app is like. Firstly, the screens are white with the signature green button. Each view has a skip button which may be pressed at any stage to fast forward to the login screen. This is very important because if the user has no way to skip past these screens, they may be more inclined to just exit the app. There is a page controller (the circles) which is used to indicate to the user what page they are on. This is very important for an onboarding experience as it signifies to the user how long is left for the onboarding. If the user knows that there are only three views until the end, then they will be less likely to skip it.
These three onboarding screens only appear the first time the user opens the app. This way, the user will only see it when they are first signing up, or if they had previously deleted the app.

## Map View (Location Sharing & Search)

An additional feature that adds to the complexity of the app is the map view. On the map view there is a search bar in which specific locations may be searched and the map will be repositioned to show you that area. The app also displays the users current location. The map view is filled with dummy user dogs, which replicate the users that would have been retrieved from the database, had the correct database been used. More detail on this in the Challenges section.

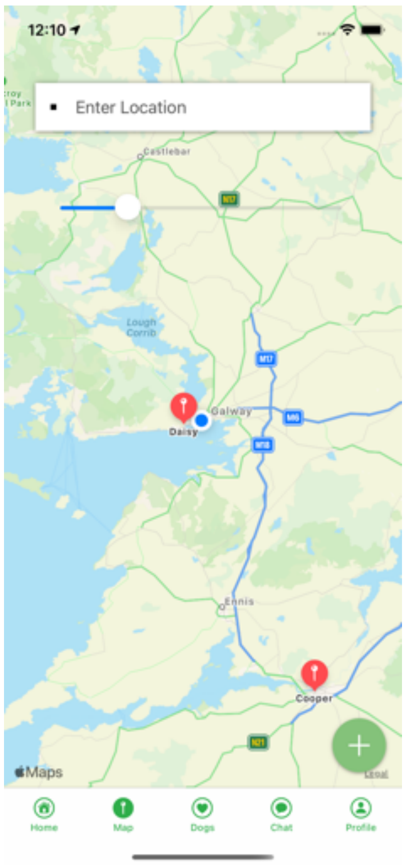


*Figure 18: Map View*

Card View (i.e., Swiping)

The ability to swipe through a deck of cards which contained dogs in them, similar to the Tinder functionality, was implemented.

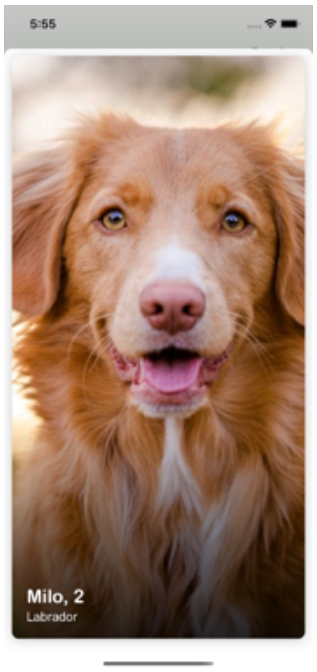


*Figure 19: Dog Card View*

### 2.3. **Non-Functional Requirements**

Non-functional requirements are aspects of the system which are necessary to incorporate, but which do not directly refer to specific user stories developed to define the system. They are general qualities which need to be borne in mind as specific functional requirements are implemented. Examples of non-functional requirements are:

*Usability*

Each aspect of the user interaction needs to be simple, but cover a comprehensive set of possibilities. The user experience should be pleasant so that users are not deterred from using the app. User experience was considered greatly in terms of the design of the application, which will be discussed in more detail in the next chapter.

*Reliability*

Also related to the user experience is the reliability of the app. It needs to be available when needed, and do what it is supposed to do. This requires extensive testing under a wide range of possible situations, including some which might appear unlikely. Edge cases were tested in the app to ensure reliable function still works. For example, if there are no dogs in the database when a Dog Minder is searching for one, the app does not crash.

*Performance*

The role performance has in an app is vital for the app's success. An app is expected to meet the user's expectations in terms of performance, or else it will deter a lot of users from continuing to use it. To ensure performance was as good as possible, only the libraries that needed to be implemented were implemented. The larger the project, the slower it would run, so libraries needed to be carefully considered before implementation. However, even still, the performance of the app cannot be judged properly when running on the simulator, as the Intel based MacBook Pro used is not very powerful.

*Maintainability*

If the maintainability of the code is not kept under consideration, it can become impossible or very expensive to change it in any way, whether for adding new features or fixing bugs.

In this system, the project source code is kept in a version control repository (Git), so is straightforward to share with any additional developers. Changes committed to the repository are annotated clearly to explain what they do.

The source code contains comments to explain the thinking behind any code which is not obvious in its function. Additionally, there is a text file outlining the structure of the app to serve as a starting point for understanding where changes might need to be made.

*Security*

The security of a system is a key requirement as it ensures protection against vulnerabilities and threats. The use of Firebase's verification and authentication service meant that security of the application is left to Google, a trusted provider.

Another form of in-house security that was implemented was that, as a requirement, users must be friends with each other in order to exchange messages. This is in a bid to prevent unsolicited messages.

Sensitive data like emails and passwords are kept private and not shown anywhere in the app.

*Scalability*

No limitations should be imposed on the app which could turn out to be bottlenecks when it is scaled to have a larger number of users or be deployed over a wider geographic region. Any assumptions the system makes should be reasonable in all situations, otherwise significant portions of code might have to be rewritten and retested later, potentially at greater expense than if they had been anticipated during the initial development.

The basis of a fully scalable app is developed already. The Firestore database used allows for large projects to be scaled. The system is designed in a manner which will allow for it to be scaled to a larger cohort of users, if necessary. Any pieces of code which refine the scale of the app, e.g. the county search can be rectified easily if necessary as it is implemented as part of a view and therefore distinct from the model and controller.

The website (which will be discussed in chapter 6), is hosted professionally therefore it can cope with all likely levels of traffic. It may also be deployed to a higher capacity service if necessary for scaling. The necessary foundation of the website is already implemented so there is scope for including additional features without undue effort.

## 2.4. **Use Cases**

Use cases describe the expected behaviour of users while carrying out a specific task. The following use cases capture the system interactions of the Madra agus Cairde App.

<u>Use Case #1 (Using the Website)</u>

Goal: Person wants to look at/use the Madra agus Cairde Website

Actor: Any person

Steps:

1. Visit https://madraaguscairde.ie/ (or the non secure version – will switch to the secure connection)
2. View website contents, i.e., look at app demonstration
3. Send contact form with any questions/concerns

<u>Use Case #1 (Registration)</u>

Goal: Person would like to register an account for Madra agus Cairde

Actor: Any person

Steps:

1. Register with email and password
2. Accept verification email
3. Login using email and password

<u>Use Case #2 (Onboarding & User Set Up)</u>

Goal: User wants to finish their initial account set up for Madra agus Cairde

Actor: Dog Owner, Dog Minder

Steps:

1. Choose account type
2. Fill in user details
3. Upload a photo of their dog and/or dog

<u>Use Case #3 (Profile - Dog Minder)</u>

Goal: Set up profile

Actor: Dog Minder

Steps:

1. Edit profile

2. Fill in details

3. Upload avatar/more pictures

<u>Use Case #4 (Dog Searching - Dog Minder)</u>

Goal: Search for dogs on Madra agus Cairde

Actor: Dog Minder

Steps:

1. Swipe through dog cards

- Left to skip over the dog

- Right to send a friend request

Or

1. Search for dogs based on location

2. Choose county

3. Press search button

<u>Use Case #5 (Map View - Dog Minder)</u>

Goal: Use the map view to see dogs nearby/search on the map to check out different locations

Actor: Dog Minder

Steps:

1. Move around on map to see pinpoints nearby

2. Use the search box to zoom into a specified location, e.g., Limerick, to see the dogs near there

<u>Use Case #6 (Profile - Dog Owner)</u>

Goal: Set up profile

Actor: Dog Owner

Steps:

1. Edit profile

2. Fill in details

3. Upload avatar/more pictures

<u>Use Case #7 (Friend Requests - Dog Owner)</u>

Goal: Accept/reject friend requests

Actor: Dog Owner

Steps:

1. Click onto request
2. Scan the profile
3. Accept/reject request
4. If accepted, send a message

Use Case #8 (Messaging Service)

Goal: Send messages – including picture message

Actor: Dog Owner, Dog Minder

Steps:

1. Type message
2. Send message
3. Attach a picture message

### 2.5. **Data**

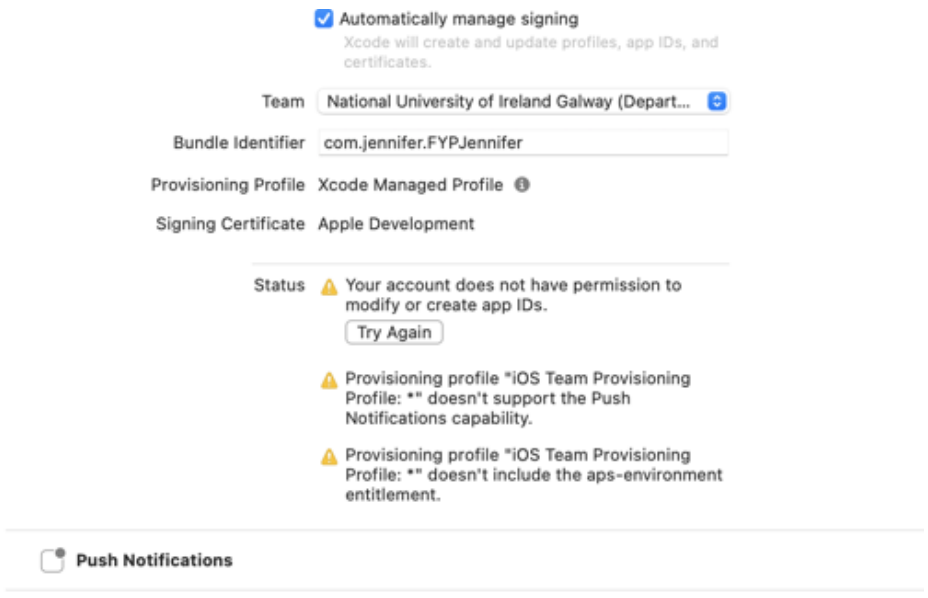
A detailed description of the data required for this app is given elsewhere in this document, but essentially the app needs basic information about the two types of person (Dog Owner and Dog Minder), the dogs, and interactions between them (e.g. Friendships and Messages).

### 2.6. **Constraints**

In developing this project, certain constraints limited what could be achieved.

*Cost and resources:*

A wide variety of development tools are available for producing iOS apps, most of which are free. However, membership of the Apple Developer Programme is required for publishing and sharing apps, as well as having access to advanced capabilities within the app. When beginning this project, a membership to the iOS Developer University Program was acquired through NUIG. This was expected to have the same capabilities as the professional version of the Apple Developer Programme, the only difference being the price. Unfortunately, however, it was discovered to be as limited as the free solo developer program. This restricted the project greatly as advanced capabilities such as Push Notifications could not be implemented, unless the USD 99 fee (annual) was paid. The decision was made to avoid this cost and therefore this feature could not be implemented.



☑ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: National University of Ireland Galway (Depart...

Bundle Identifier: com.jennifer.FYPJennifer

Provisioning Profile: Xcode Managed Profile

Signing Certificate: Apple Development

Status:
⚠ Your account does not have permission to modify or create app IDs.

⚠ Provisioning profile "iOS Team Provisioning Profile: *" doesn't support the Push Notifications capability.
⚠ Provisioning profile "iOS Team Provisioning Profile: *" doesn't include the aps-environment entitlement.

*Figure 20: Push notification permission denied*

*Scope:*

This was a major constraint in terms of developing the application. It is easy to get carried away and be overly ambitious with the features you want to implement; however, it is arguably more important to focus on a smaller scope and do it well. This project was a balancing act of developing new and exciting features to show off a more diverse skill set, and ensuring the solid core requirements were done to a high standard. The scope did have to be narrowed as it was discovered that some expectations had been too unrealistic. The list of requirements was carefully ordered and prioritised in a way that anything cut from the project would not negatively impact the project as a whole. It was also ensured that, where possible, development techniques that would have been used to develop certain features that were cut, were already visible elsewhere in the program. For example, the review section could not be implemented as time would not allow for it, but the underlying logic would have been similar to the message function because it links pairs of users together.

*Quality:*

Ensuring a high quality and standard, particularly in the UI and UX, was always going to be a tight constraint. As this project required the full development of the app i.e., front and back end, it was expected from the beginning that getting a quality product overall would be tough. It was important to dedicate a good amount of time into the front end, however, this would be worthless if the backend was completely neglected.

*Risks:*

Many risks were involved as this project was such a large undertaking, especially for one person. Running out of time, being unable to get certain features working and neglecting other modules/exams while trying to focus on getting this project done were the main risks that needed to be kept in mind. The best way to minimise risk is through the use of good project management strategies. If things are researched and planned out, then it will be harder to misjudge them. Obviously, some risks are unavoidable, so where possible, extra time should be found in the timeline and allocated for contingency.

*Time:*

The main constraint this project faced was time. The workload of a final year student is very heavy so finding enough time between assignments and lectures was nearly impossible. Time was a big factor this year in particular with exams taking place after Christmas instead of before, as the Christmas holidays could not be dedicated to FYP, it needed to be dedicated to exam preparation instead.

# Chapter 3. **Design**

## 3.1. **Introduction**

This section will describe the design approaches taken for this project. It will outline the structure of the data, as well as the overall architecture of the project. It will provide a detailed description of how the user experience/user interface evolved over the course of the project.

## 3.2. **Development Method**

The development methodology used in a project is very important as it informs the developers about how a project should be split up and worked on. While all methodologies have their advantages and disadvantages, a project's nature should be adequately assessed before choosing a method. With a sole developer working on this project, finding the right development method suited to the type of work was essential. After investigating a number of methods, it was decided that the best approach to this type of development, bearing in mind the unpredictability of the work (due to inexperience), would be an agile approach. The agile method involves developing solutions for an evolving list of requirements informed by end user feedback. This seemed like the perfect approach for this project as it was going to be heavily influenced by user testing. As there was only one developer, the decision was made to avoid sprints and stick to a Test-Driven Development (TDD) type of approach where a single feature would be worked on until it passed its tests. A feature would be verified through a number of manual tests, including edge case testing, to ensure that it had been built solidly. Soft deadlines were enforced, informed by earlier planning. Sticking to a loose timeline was difficult but important as it would be very easy for feature creep to ruin the project.

## 3.3. **System Architecture**

This app consists of an iOS application communicating with a cloud-based database, Google Firestore. Before detailing the architecture, the design decisions around choosing the iOS platform will be summarised.

*Platform*

Having used iOS devices and apps for a few years, it was felt that their familiarity would make it easier to design and develop an app for that platform. iOS encourages an intuitive user experience and overall, aesthetically pleasing appearance. After doing some research and comparing iOS and Android, iOS also seems to come out on top for a lot of reasons.

These include, but are not limited to:

- Apple is a closed ecosystem which allows developers a lot more control and stability with their apps.

- Most iPhone users stay up-to-date with iOS updates so it is also easier to guess what most users will see/what my app will be able to do on their phones.
- There are themes and design principles adhered to by all iOS apps which make them stand out and maintain their integrity and standards above those of other operating systems' apps.

For these reasons, the app was built and targeted for iOS.

*Model-View-Controller (MVC)*

A widely used software design pattern is Model-View-Controller (MVC). It separates three aspects of the program to keep them easier to understand and permit replacement of one without requiring changes to the other two.

Model is the internal representation of the data. View is the presentation layer; how information is presented to the user or obtained from the user. Controller is the code implementing the logic of the application, taking data from the model to pass to the view after any required processing, and passing any changes to the data back to the model.

If one of these areas needs to be updated, it can be done independently, as long as the pattern has been followed correctly. This offers greater flexibility in terms of device independence, for example. Devices with different display types or different update mechanisms can be accommodated with the same data model and controller. Similarly, different data stores can be used with the same view and controller.

In Swift, it is relatively straightforward to implement MVC because it stores views as one type of object, models as another and controllers as yet another class.

In the project, the storyboards hold the views. A combination of manual and programmatic design was used to create the views.

The controllers are the Swift files that implement the logic of the app, such as the

The model of the project are files that are linked to the database, for example, the FirebaseUser file.

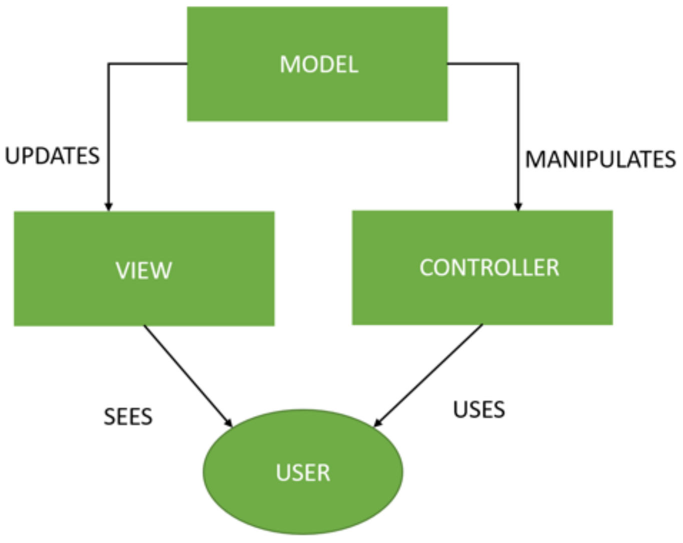


*Figure 21: MVC Diagram*

*Firebase*

Google Firebase was selected for hosting the application. Firebase offers a multitude of services under the umbrella of a single platform. As well as hosting, it was used as the database, file storage and authentication provider for the app.

Firebase keeps data in sync across client apps through real time listeners and offers offline support for mobile and web so it can be used to build responsive apps which work regardless of network latency or internet connectivity.

Firebase is a very convenient way to build a secure authentication system. It provides support for email and password accounts, as well as social logins. It deals with the necessary requirements for email and password accounts, such as email verification and error handling, e.g. checking if a password is strong enough.

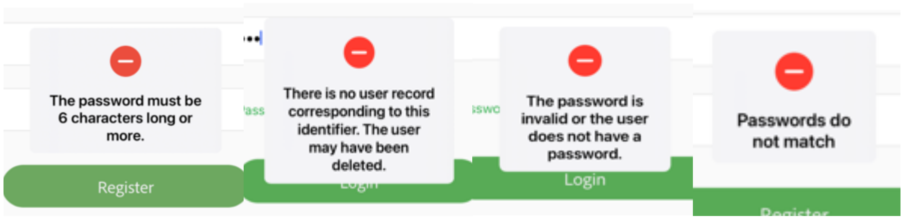
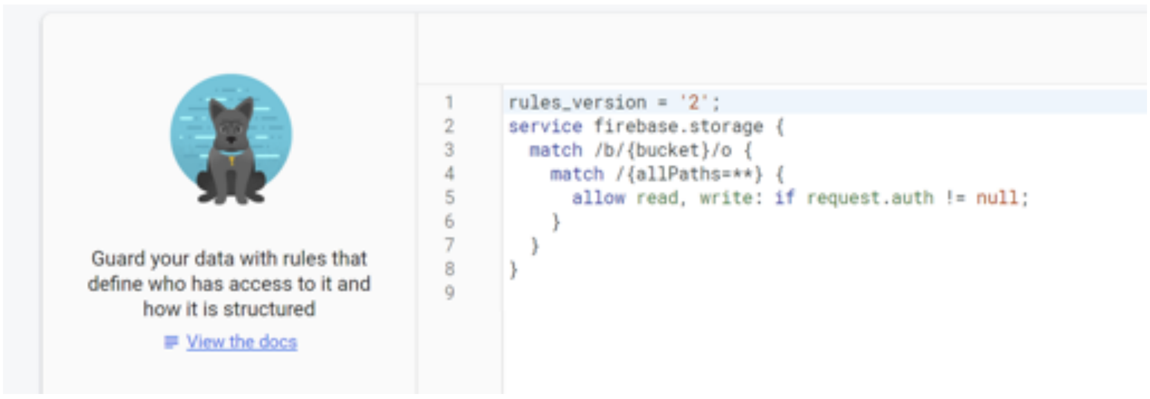

The password must be 6 characters long or more.
There is no user record corresponding to this identifier. The user may have been deleted.
The password is invalid or the user does not have a password.
Passwords do not match

*Figure 22,23,24,25: Error Handling by Firebase*

This application implemented the email and password account style login, and efforts were made to implement the Google sign-in feature. However, the latter feature was not made to work correctly by the deadline. Nevertheless, it still appears on the login screen as an example of the social login style that could be implemented in future.

Firebase Storage was used to store images uploaded by users for their profiles. Users could choose to upload pictures from their device if they allowed the necessary permissions, or they could use their camera to take a photo in real time while using the app.

The following rule was added to Firebase Storage to add further protection by restricting access to authenticated accounts.

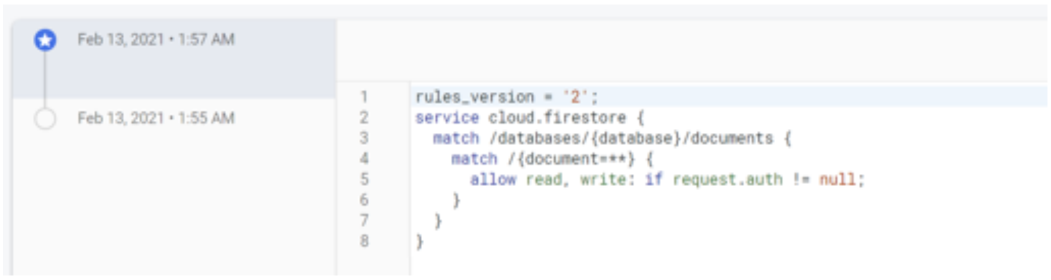

Guard your data with rules that define who has access to it and how it is structured

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

*Figure 26: Storage Rules for the Firebase Storage*

For the application's database, the Cloud Firestore was selected to store necessary information. Cloud Firestore was chosen over Firebase's other database option, Realtime Database, as early research seemed to suggest it was the more appropriate option of the two. It was discovered too late into the project that the Realtime Database, had it been implemented, would have made it possible for the map feature to work properly. Had better investigation been done in the beginning, this flaw in decision making could have been avoided. However, prior to beginning the project, the author had very little understanding of mobile development and databases, and naivety clouded judgement. Hindsight is a great teacher, and if future projects are to be undertaken, this mistake will not be made again.

The following rule, similar to the rule for the Storage, was added to the Cloud Firestore.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

*Figure 27: Database Rules for the Firebase Database*

The way in which the data was stored in the database is detailed further in the next section.

## 3.4. **Data**

This application is very dependent on a readily available database. It contains information about the users, some of which is supplied directly by the user when they create their profile, and the rest is gathered indirectly as a result of actions they take when using the app.

For example, when setting up their profile, a user is required to supply an email address, their name and birthday. This information can be entered through various input fields, such as text boxes and date pickers.

While using the app, the user may become friends with other users and exchange messages with them. This automatically generates new documents in the database which contain information related to the friend requests sent and accepted, i.e, Like contains information such as the date and the users involved (e.g., who sent the friend request to who) and Friend contains information about two users and their friendship, once they become friends.

The new documents also contain information related to conversations between friends, i.e., Messages contains information about the senders and contents of the message itself, Chat Room indicates the id that matches both users in the conversation, Recent contains information related to the most recent message sent, so it can be displayed before viewing the entire conversation and Typing enables the app to show that the other user is actively typing a reply.

A clearer way to visualise the layout of the database is through this entity relationship diagram:

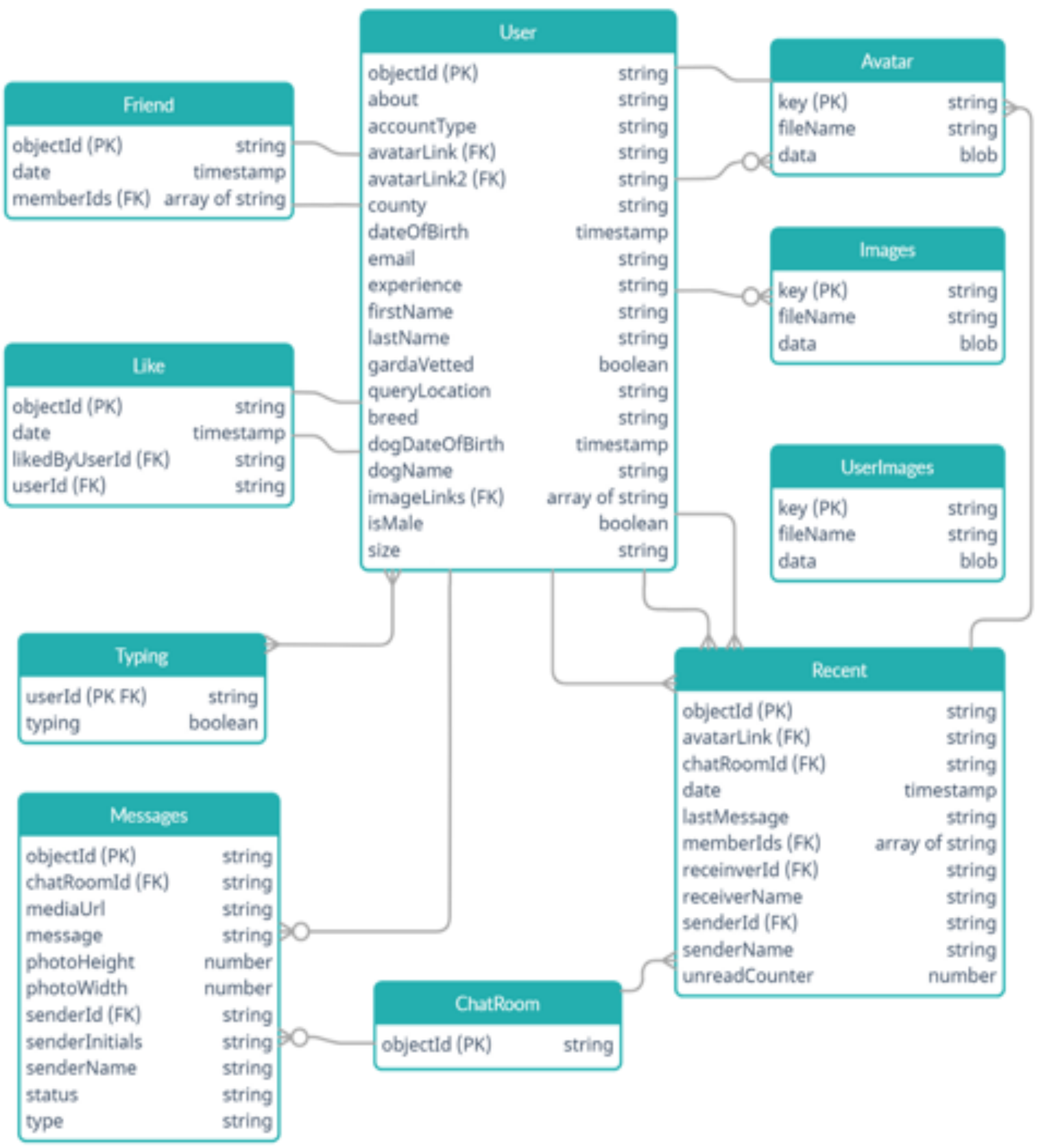
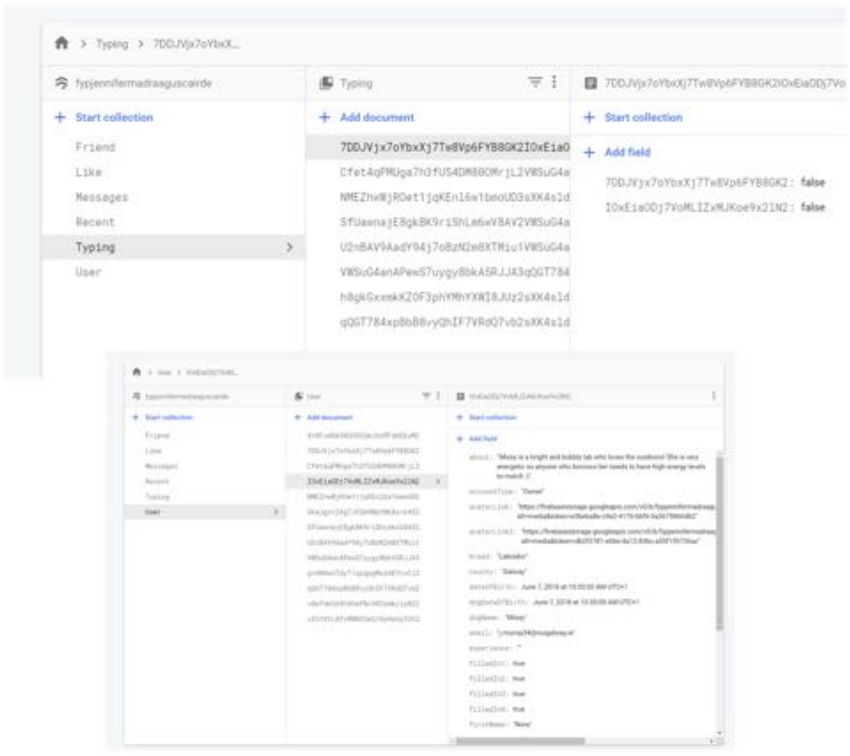


*Figure 28: Madra agus Cairde Entity Relationship Diagram*



Typing > 7DDJVjx7oYbxX...

typjennifermadraaguscairde

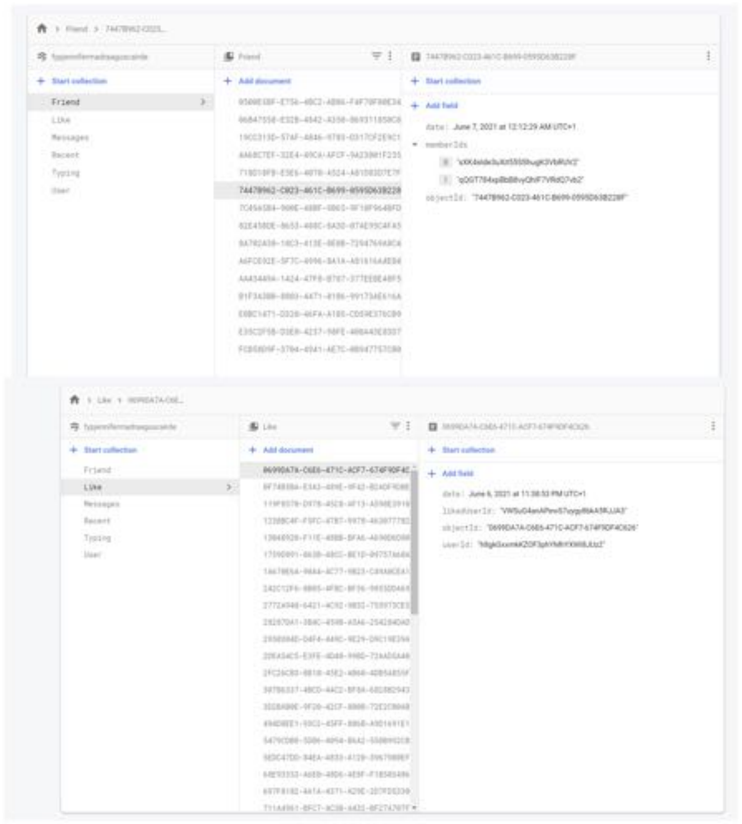
Typing

7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2IOxEia0Dj7Vo

```
Friend      7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2IOxEia0     7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2: false
Like        Cfet4qPMUga7h3fUS4DM88OMrjL2VWSuG4a     IOxEia0Dj7VoMLIZxMJKoe9x21N2: false
Messages    NMEZhwWjROet1jqKEn16w1bmoUD3sXK4s1d
Recent      SfUawnajE8gkBK9riShLm6wV8AV2VWSuG4a
Typing      U2nBAV9AadY94j7oBzN2m8XTMiu1VWSuG4a
User        VWSuG4anAPew57uygy8bkA5RJJA3qQGT784
            h8gkGxxmkKZ0F3phYMhYXWI8JUz2sXK4s1d
            qQGT784xpBbB8vyQhIF7VRdQ7vb2sXK4s1d
```

```
about: "Missy is a bright and bubbly lab who loves the outdoors! She is very energetic so anyone who borrows her needs to have high energy levels to match :)"
accountType: "Owner"
avatarLink: [illegible]
avatarLink2: [illegible]
breed: "Labrador"
county: "Galway"
dateOfBirth: June 7, 2018 at 10:35:00 AM UTC+1
dogDateOfBirth: June 7, 2018 at 10:35:00 AM UTC+1
dogName: "Missy"
email: [illegible]
experience: ""
filledIn1: true
filledIn2: true
filledIn3: true
filledIn4: true
firstName: "None"
```

Figure 29: Structure of Typing and User in the Database



Friend > 74478962-C023...

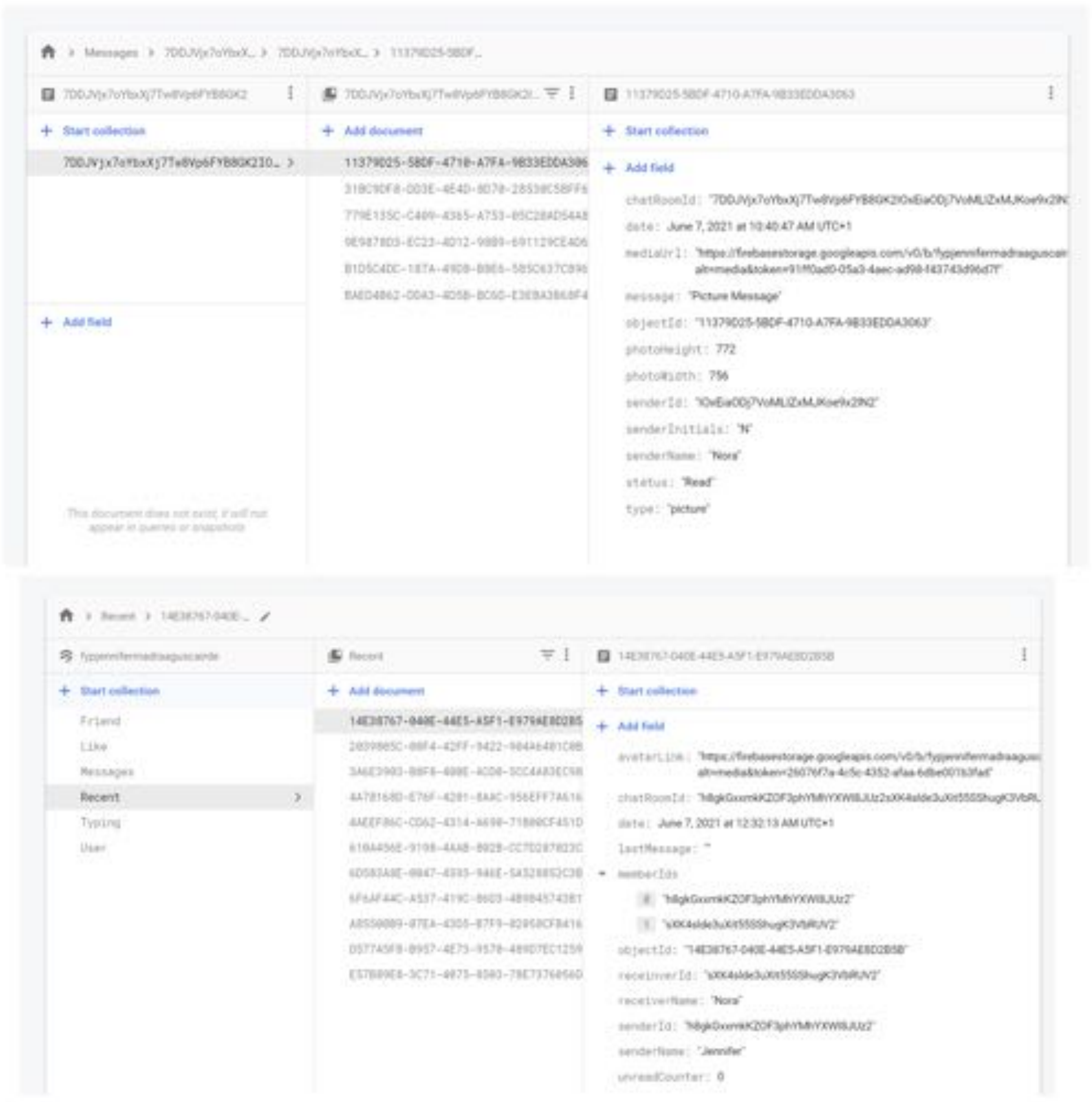
74478962-C023-461C-B699-0595D63B228F

```
date: June 7, 2021 at 12:12:29 AM UTC+1
memberIds
    0 [illegible]
    1 "qQGT784xpBbB8vyQhIF7VRdQ7vb2"
objectId: "74478962-C023-461C-B699-0595D63B228F"
```

Like > 8699DA7A-C6E6...

```
date: June 6, 2021 at 11:38:53 PM UTC+1
likedUserId: [illegible]
objectId: "8699DA7A-C6E6-471C-ACF7-674F9DF4C626"
userId: [illegible]
```

Figure 30: Structure of Friend and Like in the Database


```
Messages > 7DDJVjx7oYbxX... > 7DDJVjx7oYbxX... > 11379D25-5BDF...

7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2
+ Start collection
7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2IO...
+ Add field
This document does not exist, it will not appear in queries or snapshots

7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2I...
+ Add document
11379D25-5BDF-4710-A7FA-9B33EDDA306
31BC9DF8-DD3E-4E4D-8D70-28538C58FF6
779E135C-C409-4365-A753-05C28AD54A8
9E987BD3-EC23-4D12-9889-691129CE406
B1D5C4DC-187A-49D8-B8E6-5B5C637C896
BAED4862-DD43-4D5B-BC6D-E3EBA3868F4

11379D25-5BDF-4710-A7FA-9B33EDDA3063
+ Start collection
+ Add field
chatRoomId: "7DDJVjx7oYbxXj7Tw8Vp6FYB8GK2IOxEiaODj7VoMLIZxMJKoe9x2IN
date: June 7, 2021 at 10:40:47 AM UTC+1
mediaUrl: "https://firebasestorage.googleapis.com/v0/b/fypjennifermadraaguscair
          alt=media&token=91ff0ad0-05a3-4aec-ad98-f43743d96d7f"
message: "Picture Message"
objectId: "11379D25-5BDF-4710-A7FA-9B33EDDA3063"
photoHeight: 772
photoWidth: 756
senderId: "IOxEiaODj7VoMLIZxMJKoe9x2IN2"
senderInitials: "N"
senderName: "Nora"
status: "Read"
type: "picture"
```

```
Recent > 14E38767-040E...

fypjennifermadraaguscairde
+ Start collection
Friend
Like
Messages
Recent
Typing
User

Recent
+ Add document
14E38767-040E-44E5-A5F1-E979AE8D2B5
2839985C-08F4-42FF-9422-984A6401C08
3A6E3903-B8F8-400E-ACD0-5CC4A83EC98
4A78168D-E76F-4281-8AAC-956EFF7A616
4AEEFB6C-CD62-4314-A698-71B80CF451D
610A456E-9198-4AAB-8928-CC7D287823C
6D583A8E-0847-4393-946E-5A528852C38
6F6AF44C-A537-419C-8603-48984574381
A8550089-87EA-4305-87F9-82958CFB416
D577A5F8-8957-4E73-9578-489D7EC1259
E57B89E8-3C71-4073-8583-78E73760560

14E38767-040E-44E5-A5F1-E979AE8D2B5B
+ Start collection
+ Add field
avatarLink: "https://firebasestorage.googleapis.com/v0/b/fypjennifermadraagusc
            alt=media&token=26076f7a-4c5c-4352-afaa-6dbe0010b3fad"
chatRoomId: "h8gkGxxmkKZOF3phYMhYXWI8JUz2sXK4slde3uXrt5SSShugK3VbRL
date: June 7, 2021 at 12:32:13 AM UTC+1
lastMessage: ""
▾ memberIds
    0  "h8gkGxxmkKZOF3phYMhYXWI8JUz2"
    1  "sXK4slde3uXrt5SSShugK3VbRUV2"
objectId: "14E38767-040E-44E5-A5F1-E979AE8D2B5B"
receiverId: "sXK4slde3uXrt5SSShugK3VbRUV2"
receiverName: "Nora"
senderId: "h8gkGxxmkKZOF3phYMhYXWI8JUz2"
senderName: "Jennifer"
unreadCounter: 0
```

*Figure 31: Structure of Messages and Recent in the Database*

According to the General Data Protection Regulation (GDPR), only information needed for the app should be collected and maintained. Therefore, Madra agus Cairde made an effort to collect as little information about the users as possible, to preserve their privacy.

While the surname is not being displayed to other users, it is a safety precaution to know everyone's true identity in case of safety concerns arising.

The first name and age of users is displayed on their profiles and it helps to personalise their profiles and give other users enough information about them. The age of someone may play a factor into whether a Dog Owner is happy to let them mind their dog or not. There is currently no age restriction for the app, however, further down the line, if any safety concerns were to arise, an age limit of either 16+ or 18+ would be implemented.

## 3.5. UX/UI

For applications in general, but mobile apps in particular, the user experience (UX) is a vital component. The user interface (UI) is a fundamental part of the UX. Without a meaningful user experience, an app will not be successful.

A particular operating system, like iOS, adopts its own particular style of user interaction which needs to be respected by developers creating apps for it. Otherwise, the app will not "feel" right and risks being rejected by users as difficult to use. Some of the style will arise automatically by making use of operating system resources, while some app-specific operations need to be implemented in a way consistent with the way other apps implement comparable features.

*Prototyping:*

Prototyping is a great tool for UX research. A variety of prototypes were developed for this project, ranging from low fidelity to high fidelity.

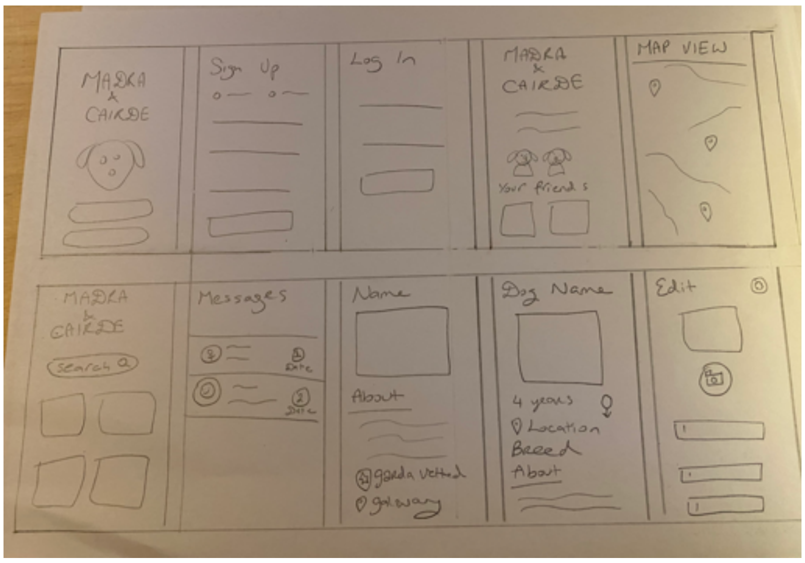


*Figure 32: Example of Low Fidelity Prototypes for Madra agus Cairde*

The figure above depicts some of the low fidelity design prototypes that were among the hand sketched drawings from the early stages of the project.

Initial concepts were inspired by apps of similar nature on websites such a Dribble.com or Behance.com. Being able to browse through many aesthetically pleasing app concepts and ideas there really helped inspire and stimulate imagination. Research was also done into other apps which have two user account types, such as Airbnb (host and guest). In terms of design and functionality, this research was important as it ensured that this app would work in a logically sound way.

There was a lot of toying around with the sketches until more solid ideas formed. Then, these we transferred to higher-fidelity prototypes, using helpful technologies such as Miro, for really minimal and grey scaled concepts.

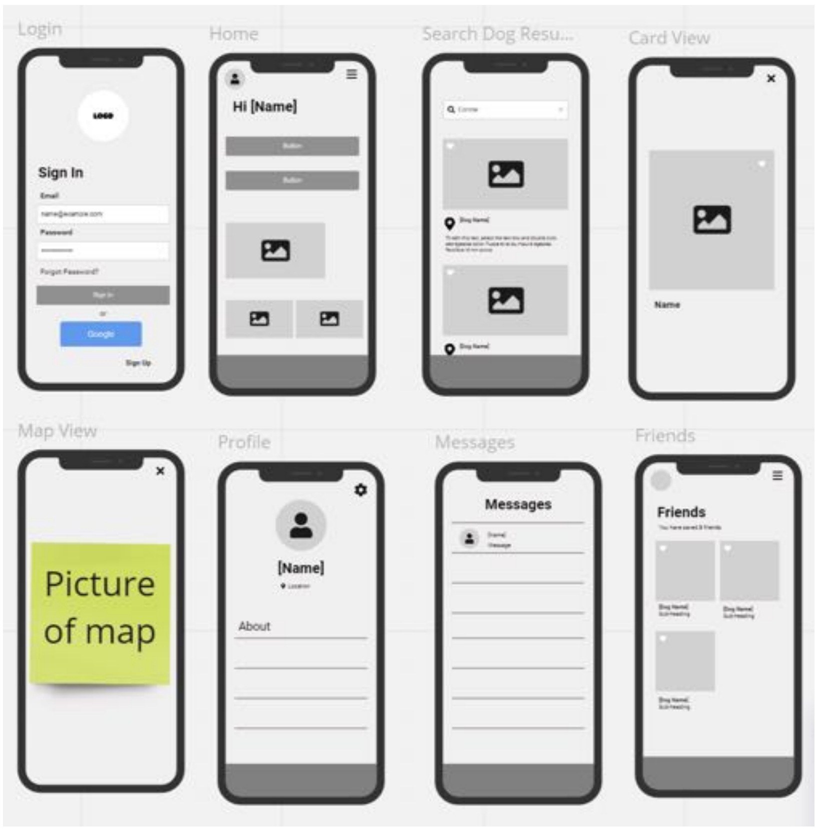


*Figure 33: Grey Scale Wireframes created in Miro*

When the idea of colour was introduced into the prototypes, Miro was switched out for Adobe XD as Adobe XD is more heavily focused on colour and more advanced prototyping components.

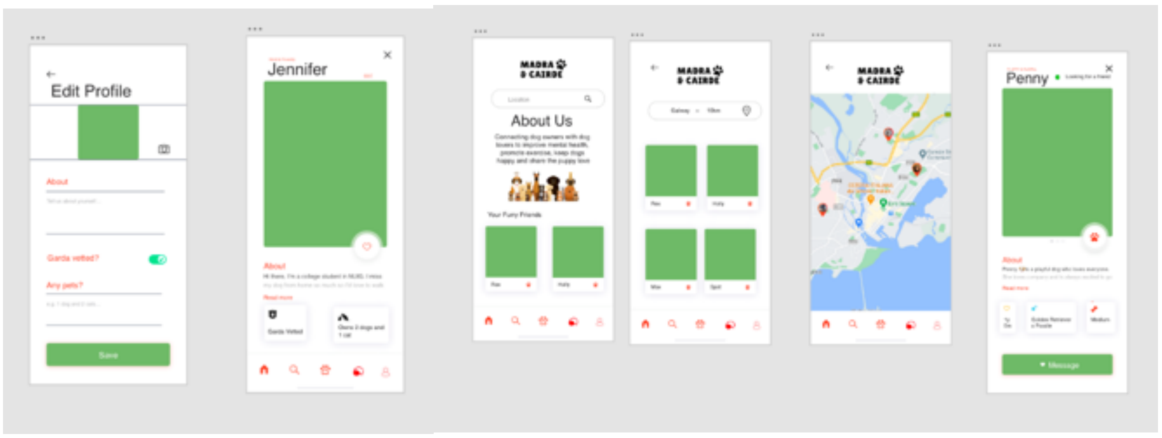


*Figure 34: High Fidelity Prototypes Designed in Adobe XD*

Using an Agile methodology to develop really helped with the transition from the hand drawn sketches (*Figure 32*) to the very basic grey scale prototypes (Figure 33), to the coloured prototypes (*Figure 34*) to what the end result is (*Figure 35 & 36*), because of the focus it puts on user feedback and input.

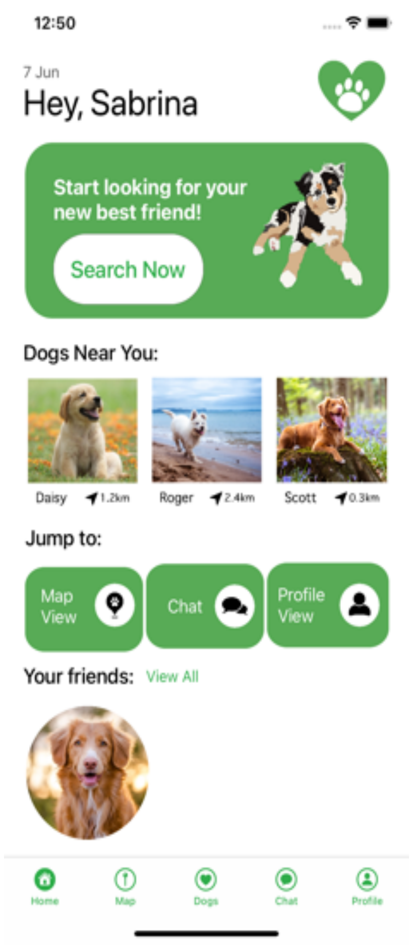


Figure 35: UI of Home screen for Dog Minder

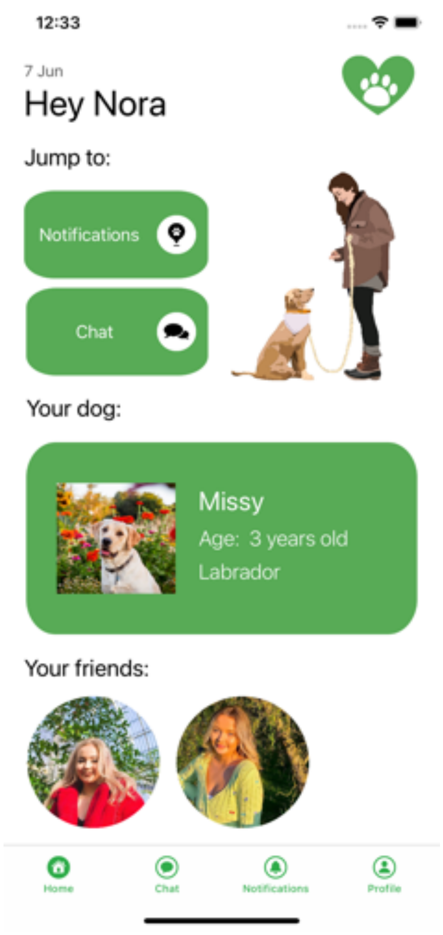


Figure 36: UI of Home screen for Dog Owner

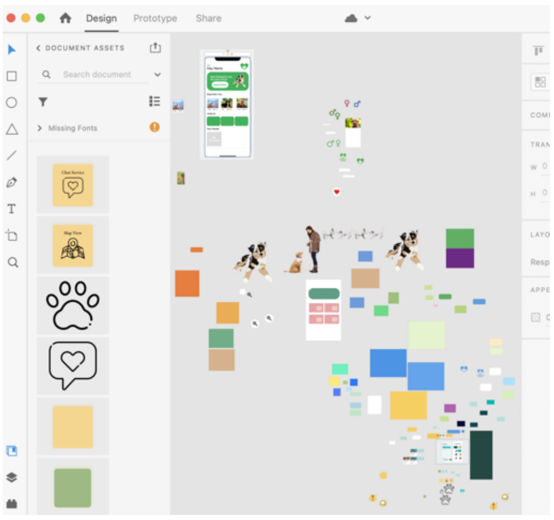


Figure 37: Example of how Adobe XD was used for making graphical components

*User Testing*

User Testing is a very important aspect of app development. It gives invaluable feedback which can be used when iterating upon a given design to improve. It can help ensure the effectiveness of a certain implementation and ensure there is a common understanding between developer and users. As a developer, oftentimes it is hard to see the pitfalls of something you have built because it may seem so straightforward to you. It is vital to get other people's perspectives to ensure that the design is as well thought out as you believe it to be.

User testing was a fundamental necessity when it came to building this project. User testing was first implemented in the early days of this project, when trying to scope out the idea for the project. A huge amount of feedback from friends and family was used, to gauge how good of an idea this was.

User testing was then used on the range of prototypes that were mentioned above. This feedback was accepted and translated into improvements in the design and flow of the app. User testing was the best way to ensure that the end product would be stakeholder feedback focused, because the user testers that were used are all perfect candidates for the app.

*Design Choices*

As much as possible, when a design choice had to be made, it was checked with a panel of potential app users. This feedback proved invaluable in coming up with an attractive design.

The appearance of an app needs to be distinct enough to make it stand out, but similar enough to other apps to make new users feel comfortable. One way to achieve this is by using particular colour palettes. There are basic design rules which apply to colour selection.

White was picked as the background colour because it provides contrast with other visual features. This makes elements stand out for all users, but particularly those with poor vision, including colour-blindness.

The main thematic colour used in the app is a shade of green. This was chosen because it mirrors the outdoor nature of the app and because it was liked by most of the testing panel. The particular shade was chosen based on a range of options shown to the panel.

The layout of each screen was kept as simple as possible, to avoid overloading the users with information and to keep it usable by people with limited vision problems.

*Graphic Elements*

CANVA pro was accessed for the duration of the project. It was extremely useful for the many assets it has.

Photoshop:

Photoshop was used for a variety of reasons during this final year project. The main reason for using it was for drawing the three pieces of art that are used throughout the app.

*Figure 38: Dog walker with dog on lead (Drawn by Jennifer Murray)*

*Figure 39: Two dogs playing (Drawn by Jennifer Murray)*

*Figure 40: Dog (Drawn by Jennifer Murray)*

3

An iPad and Apple pencil were borrowed from a friend for a week to draw these pictures. Any extra editing needed over the course of the development process (e.g., changing the colour of the dog's collar) was done in Photoshop Elements on a laptop. Being able to personally draw these images for the app was important as it would mean that the artwork would be unique and royalty free. It also ensures a cohesive aesthetic throughout the app. Had there been more time, more images similar to the above would have been drawn and integrated into the app.

These three images are displayed on the onboarding screens as well as the home page of the app.

Photoshop was also used to generate the favicon.ico for the website. (This is the icon shown by a web browser, usually on the tab associated with the site.)

An app icon generator was used to turn one of the art pieces drawn for the project into the app icon that appears on a user's device.

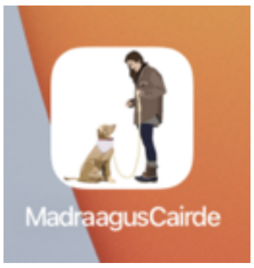


*Figure 41: Example of how Adobe XD was used for making graphical components*

*Ease of use features*

Key elements present in applications with a good user experience are what are known as 'ease-of-use' features. These features aim to remove impediments which might detract from the user experience. Operations should be as smooth as possible while having the required effect. This application strove to implement many accessible and easy to use features to help encourage a good user experience among users.

- Keyboard Types:

  Keyboards may seem like a very trivial aspect of UX design. However, they are an extremely important feature for ensuring the ease and satisfaction of a user's experience with an application. Madra agus Cairde ensures that the correct keyboard for a specific need is triggered at the right time. For example, when the user taps on the email text field when they wish to register/login, the keyboard pops up immediately, and the keyboard is set to the 'email' type, i.e., it has the @ symbol in an easy and accessible place for the user. As soon as the user taps out of a text field or a button to continue, the keyboard immediately disappears. This kind of detail orientation is widespread throughout the app.

- Input Fields:

  Another seemingly trivial aspect of UX design is the area of input fields, such as text fields. It may seem like something small, but offering a user a selection of options to choose from instead of having to type unnecessarily greatly improves user experience. This was found to be true in the number of user tests carried out for this project. A common answer among user testers was that they are often ‘lazy’ when signing up to apps and do not like filling in a huge amount of writing. This was taken on board when implementing the user details set up pages. Any answer that could be made into a selection was made so, e.g., the size of a dog. This sentiment also carried through to date pickers. The users tested preferred the look and use of the date picker, compared to having to type in their date of birth from scratch.

Obtaining the user’s actual date of birth instead of just their age was also very important for data accuracy too, as a static age will become outdated very quickly which will require unnecessary extra effort from the user to rectify.

# Chapter 4. **Implementation**

## 4.1. **Introduction**

To complete this project, an iOS mobile application must be developed. This project has a number of different aspects related to it and therefore will require a number of different components to be implemented. Implementation should fulfil the design specifications while matching the requirements, both functional and non-functional, which were mentioned in chapter 2. This chapter will outline the development tools which were used in the implementation of this app, as well as any necessary development considerations that needed to be made.

## 4.2. **Development Tools**

*iOS*

At the start of this project, with no prior knowledge of iOS development, suitable tools had to be chosen. For the language to use, the principal choice was between Swift and Objective C, although other languages could have been used, including high level ones like Delphi (Pascal). Investigation was done and Objective C appeared to be very similar to C++, which is a notoriously difficult language to read. Swift, on the other hand, appeared to more closely resemble the English language, and therefore closer to Java/JavaScript etc.

Being able to easily understand the code that would be written, preserving its maintainability, was considered very important. Swift seemed like a good choice but before final decisions were made, more research was carried out to make sure development would not be hindered by this choice. Luckily, in pretty much every aspect, Swift came up as the preferred language over Objective C; it is faster and requires a smaller amount of code.

Another early decision made was to use UIKit. Even after choosing Swift, there are options when it comes to developing an app. UIKit can be used. This is a built-in framework in iOS which implements important UI elements such as views and controls.

The alternative would have been SwiftUI, which is essentially HTML and CSS. However, previous experience with CSS (on work placement) had proved frustrating, particularly in positioning controls exactly where they needed to be. In an attempt to avoid this, UIKit's drag-and-drop approach seemed preferable.

Later on, it emerged that UIKit does have some drawbacks. Auto layout constraints proved to be quite challenging. Their job is to ensure that elements of the UI always stay in the correct position, even when switching between different iPhone models (different screen resolutions). It took a long time to get used to it and be able to correctly position elements when developing new pages.

UIKit is also not ideal for developing complex and aesthetically pleasing user interfaces. Unfortunately, all the tutorials demonstrating how to get a clean and pretty looking app tend to use SwiftUI instead of UIKit. Therefore, it made the later stages of development a lot more challenging as UIKit was not sufficiently flexible for the desired designs. Were the final year project to be restarted, it would be worthwhile learning SwiftUI and using that to define the app's visual elements.

As UIKit was the layout tool selected, it is worth noting some observations on its use. When starting an Xcode project from scratch, to use UIKit, the 'storyboard' option is picked so that the project opens with the necessary support files.

The storyboard itself is where the UI is implemented. There are many elements and tools offered by Xcode when designing the screens of the app. They allow elements to be dragged and dropped onto the storyboard to give a representation of how the app will look.

Dragging and dropping is not the only approach possible; in some cases it makes more sense to add items programmatically. In this app, examples are the search bar and zoom slider on the map view.

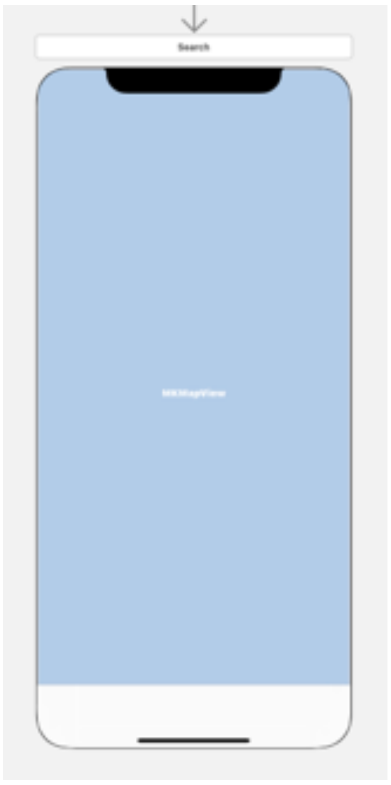


*Figure 42: View Controller of Map View*

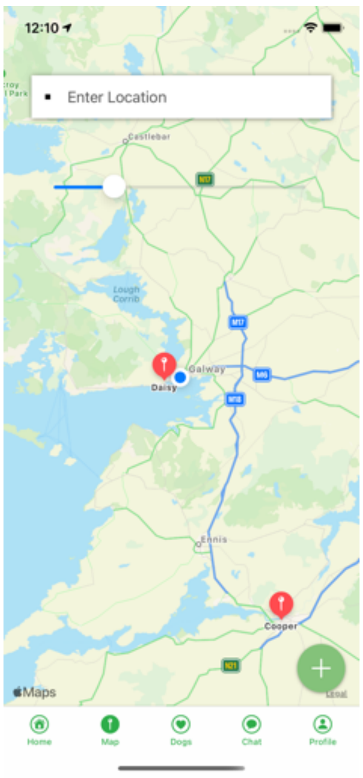


*Figure 43: Actual UI of Map View*

Using both approaches was worthwhile for the knowledge and experience.

## CocoaPods

The project utilises the CocoaPods tool. CocoaPods makes project management easier by centralising the management of third-party SDKs providing useful additions to Swift (and Objective-C) projects.

For the project, the following CocoaPods were used:

*SKPhotoBrowse*

This library allowed a user to view photos as a gallery. An example of it being used in the app is demonstrated in the figure below.

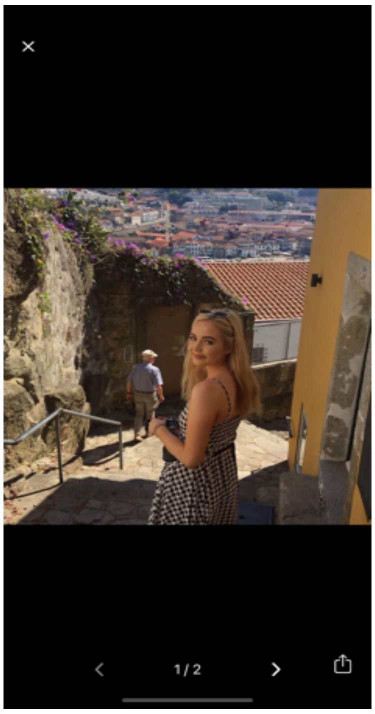


*Figure 44: Photo Browser Display for Dog Minder*

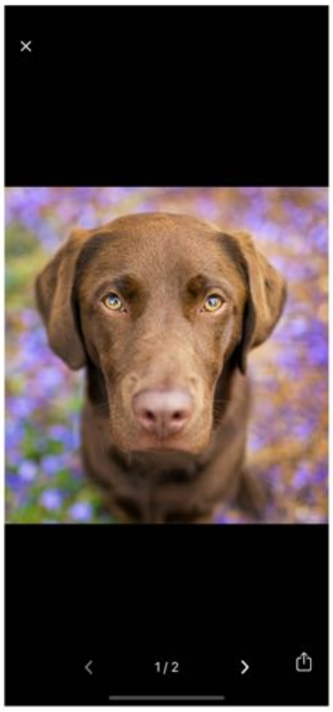


*Figure 45: Photo Browser Display for Dog Owner*

*Firebase:*

Core, Auth, Firestore, Storage, Messaging

A number of Firebase pods were required for the full use of Firebase in the application.

*Gallery*

The Gallery pod was utilised in the app whenever a user was uploading a photo for their profile. Gallery has a variety of different uses. The way it was implemented in this project was to allow a user to upload one picture as their avatar, a limit of up to 10 photos could be uploaded for the user's profile or the user had the option to use their camera to take a photo. The option to upload a picture is displayed by an alert controller that pops up after the camera button was pressed.

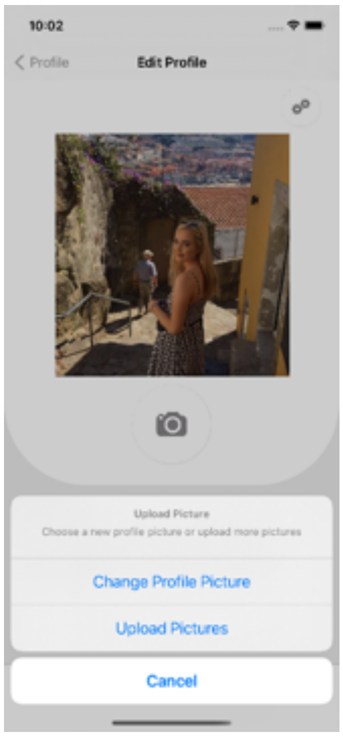


*Figure 46: Alert controller of Upload options*

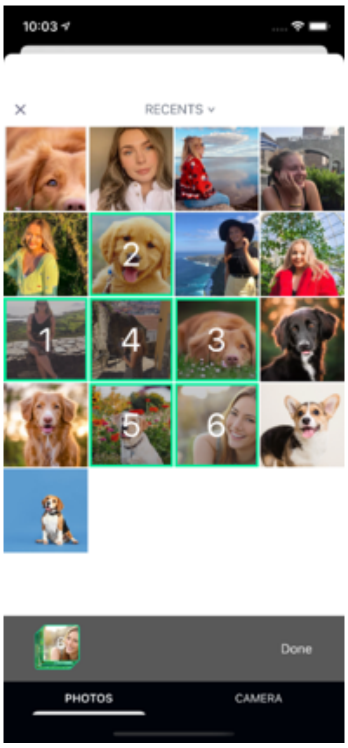


*Figure 47: Multiple Photo Selection option*

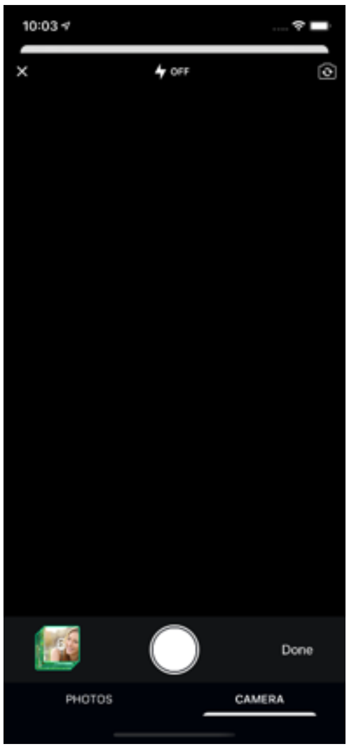


*Figure 48: Camera Option*

*NVActivityIndicatorView/AppExtension*

This library was used for displaying loading animations throughout the app, whenever the app was waiting on information from the database.

*ProgressHUD*

This was an incredibly helpful library to use during the development of this project. These progress indicators were used at various stages of the app, whenever information needed to be relayed to the user relating to progress.

*Shuffle-iOS*

This library was used in a video tutorial that was followed. The idea of a card swipe for users was popularised by Tinder and related apps. It seemed like this may be a good way to swipe through the dogs in the app and it was an alternative to the location-based search that was already implemented.

## 4.3. **Development Considerations**

Modern operating systems like iOS and Android are privacy aware, which means that users must grant explicit permission for access to particular resources. Applications which want to use these resources need to request each permission explicitly and ideally supply a reason for seeking permission. Providing a reason for the request is called “permission priming” and makes it more likely that the user will grant permission.

This project was required to seek permission for several resources; the media library, the camera, and the device location.

The application required a user to upload media for their profile picture, which meant access to their camera roll and camera was needed. This request is displayed to the user by the operating system the first time the user attempts to upload a picture. This request can be accepted or rejected; however, if it is rejected and the user wishes to upload a picture in future, the permission must be granted from the operating system settings.

For the map feature to display the user’s current location, explicit permission must be granted. This is requested by the app the first time the user clicks onto the map page. They have the option to either decline, allow only once or allow while using the app. If a user chooses to decline, then the location feature can not be used and the map of Ireland is just displayed instead of their precise location. If the user chooses to allow once, then the feature will work correctly, i.e., showing the user's current location. However, the next time the user logs into the app, it will not show their precise location again. If a user chooses the ‘allow while using app’ option, another alert is sent to the user. This prompt asks if they want to change it to allow always or keep to just while using the app.

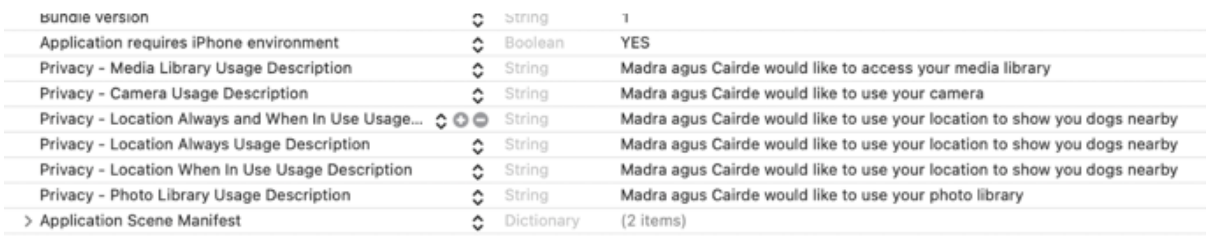
Regardless of what option is chosen by the user, my app will remember their response and the alert is not shown to the user again. If a user changes their mind in future and wants to change the permission settings they originally gave, then changes can be made in the privacy part of their iPhone settings.

For the app to use device cameras, the NSCameraUsageDescription key was included in the app's Info.plist file.

For the app to have access to a devices media library, the NSPhotoLibraryUsageDescription key was also added to the Info.plist file.

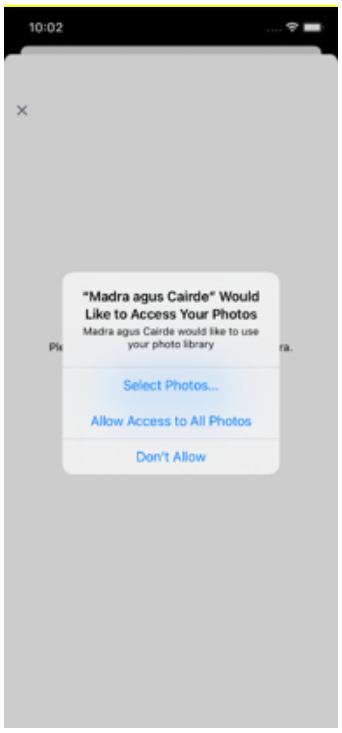
For the user’s current location, NSLocationAlwaysUsageDescription was also added.

For each key, the user is provided with a clear message in which an explanation is given as to why the app would like to have access to whichever permission it is requesting they grant the app access to.



| | | |
|---|---|---|
| Bundle version | String | 1 |
| Application requires iPhone environment | Boolean | YES |
| Privacy - Media Library Usage Description | String | Madra agus Cairde would like to access your media library |
| Privacy - Camera Usage Description | String | Madra agus Cairde would like to use your camera |
| Privacy - Location Always and When In Use Usage... | String | Madra agus Cairde would like to use your location to show you dogs nearby |
| Privacy - Location Always Usage Description | String | Madra agus Cairde would like to use your location to show you dogs nearby |
| Privacy - Location When In Use Usage Description | String | Madra agus Cairde would like to use your location to show you dogs nearby |
| Privacy - Photo Library Usage Description | String | Madra agus Cairde would like to use your photo library |
| > Application Scene Manifest | Dictionary | (2 items) |

*Figure 49: Info.plist which displays permission requests*



"Madra agus Cairde" Would Like to Access Your Photos

Madra agus Cairde would like to use your photo library

*Figure 50: Example of permission request from the OS*

# Chapter 5. **Code Quality**

## 5.1. Introduction

The quality of code is a very important factor when it comes to software development projects. Having a high quality of code will make the project more sustainable and maintainable. It minimises a build-up of technical debt and makes continuous development in the project much easier. This chapter will detail how a good standard of code was maintained during this project. The use of testing, both user and system, as well as continuous integration/continuous development (CI/CD) will also be discussed.

## 5.2. User Testing

User testing is the process of evaluating the design and user experience of a system. This is done by getting participants to try it out. The people involved in user tests must be separate from those involved in the development. The developers can of course have an opinion and be involved in their own personal testing of the application, however, for proper user testing, it is important to get unbiased opinions. User testing requires people to be able to speak their mind, so that the results can be accurate and reliable. This is important as results from user tests are often used as justification for certain design choices.

User testing has been implemented at every stage of the development process of this app. User feedback from these tests was what paved the path for the design and flow of this app. This has already been described in the Design chapter (chapter 3). Once the main development work of the app had been completed, user testing was conducted to see how a user would respond to the app as a whole, not just a subsection of it, e.g a single view or colour scheme.

Time was set aside in the project management plan to allow for changes to be implemented based on the final user testing results. The end-stage user testing was important to gain an understanding of the overall experience that the users feel.

Alternative arrangements had to be made for most user tests (bar the several people who were living in the same house as the developer). There was a relocation during the development process, which did have the advantage of being able to do in-person tests with more users.

Users who took part were asked to speak aloud as they navigated through the app. They shared their opinions about certain choices and explained their likes and dislikes. They were asked to describe the overall feel of the app and what improvements (if any) they would like made.

The alternative tests that were held among people who were not in the developer's bubble, were captured through Zoom. As they did not have access to the app, the developer came up with another way to test them, which would provide invaluable feedback. These users were specifically testing the workflow of the app. Meaning, during testing, a question would be asked in relation to specific buttons or symbols. The applications workflow and design could be deemed successful if the user tests could prove that every symbol could be correctly identified and if every feature or control is correctly interpreted.

A lot of very useful information was gained during the user testing of the app. These results were taken on board and greatly contributed to the finished product. User testing helped ensure that the design was stakeholder feedback focused.

### 5.3. System Testing

System testing is the process of searching for software errors or bugs. The goal of system testing is to actively search for bugs so solutions to fix them can be put in place. System testing can be achieved in a number of ways, for example, through scripts, but another good way to discover bugs is to manually use the app, testing normal features and edge cases, to see if any bugs can be noticed. System testing really enhances the quality of a system, as bugs that may have otherwise been left unnoticed can be noted and fixed.

A large number of system tests were done on a regular basis, on new features and in some cases, the app as a whole was testing to make sure no new bugs were introduced. Any errors or bugs were noted by the author and efforts to resolve these bugs were taken.

The table below depicts the type of bugs that were caught during system testing. The status of the bugs is denoted by their key (the legend describes the meaning of each key).

The majority of bugs were managed to be resolved by the deadline, however some may have been more complicated and time consuming to resolve properly, so a work around was put in place. This means that a course of action to stop a certain bug from happening was put in place, however this solution is not the best one and could be improved, which the author admits. Some errors represented features that were incomplete and this is why they were left unfixed.

| ***Key*** | |
|---|---|
| ✅ | **Fixed** |
| ✓ | **Work-around made** |
| ❌ | **Not fixed** |

| ***Issue*** | ***Sta*** |
|---|---|

| | ***t u s*** |
|---|---|
| **Login Page** | |
| Google Sign in not hooked up properly | ❌ |
| **Home Page** | |
| Display dog picture on home page not working | ✅ |
| Not going to correct home screen after login | ✓ |
| Buttons to pages does not change the tab bar item to the correct one (i.e., clicking map view button will still show that you are on ‘home’ on the tab bar) | ❌ |
| Not displaying today's date correctly | ✅ |
| **Tab Bar** | |
| Tab bar items are not desired colour | ✅ |
| **Chat View** | |
| Tab bar items are not desired colour | ✅ |
| Back button not available from the chat screen | ✅ |
| **Card View Page** | |
| Empty View when all cards are swiped | ❌ |
| Cannot click into profile from card | ✅ |
| **Dog Profile Page** | |

| | |
|---|---|
| Age not displaying properly | ✓ |
| Showing age of the owner, not the dog | ✓ |
| Asks for location when uploading a picture | ✓ |
| **Search Page** | |
| Dogs not showing when search button is pressed | ✓ |
| Where field for location not working | ✓ |
| **Map View** | |
| Display dogs on map view | ✓ |
| Polyline not being drawn but pinpoint is appearing | ✗ |
| **Website** | |
| Not going to the secure part of the website | ✓ |
| Footer not staying at the bottom of the website | ✓ |
| **Misc** | |
| If there's no picture uploaded then it looks bad – needs a placeholder picture | ✓ |
| Keyboard not showing correctly | ✓ |

## 5.4. **Continuous Integration/Testing**

Continuous Integration is a widespread practice in software development. It is a practice used when programmers are developing and integrating code into a repository. It ensures a consistent working version is always maintained on the main branch in the repository.

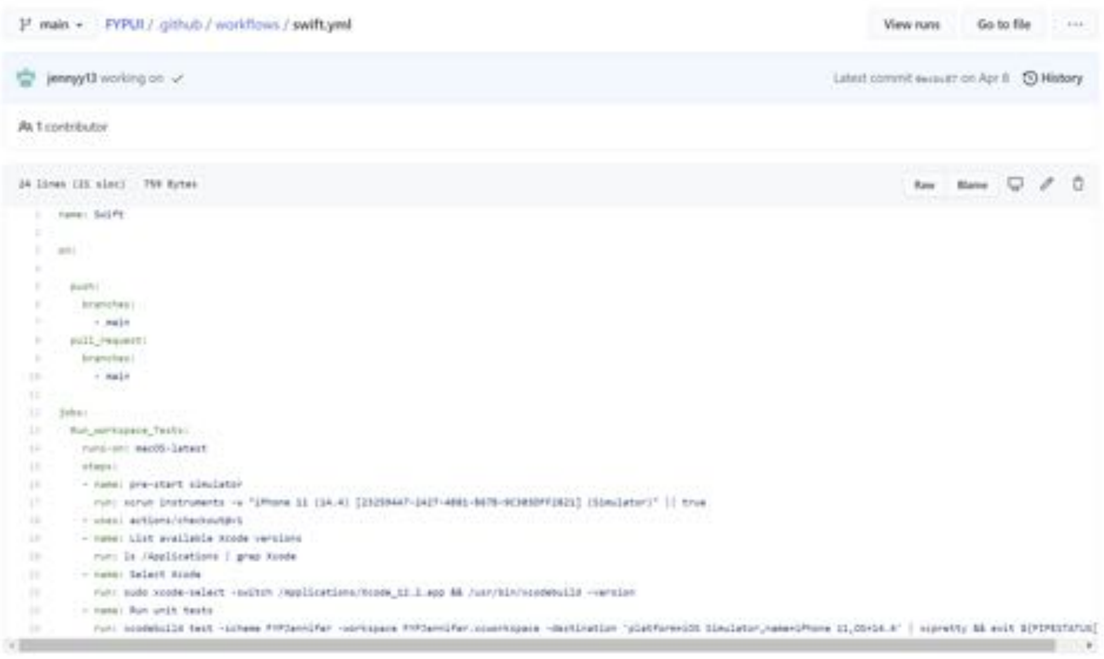
To ensure broken code does not slip through and cause the main branch of code to fail, any attempts at integration are verified by tests and builds that are automated upon any pull requests (PR).
The importance of continuous integration and development was learned while on professional placement in third year. It is vital – particularly in cases where multiple people are working on the same code – that automated testing and builds occur to ensure the integrity of the main branch.

To implement continuous integration into the project, GitHub actions was used to set up an automated test that must past in order for a PR to be merged. Branch rules were also set up to make it impossible for direct pushes to main. Therefore, to integrate new code onto the main branch, a separate branch must be pushed up and it must pass the automated test (i.e., have no failures on it).

To set this up, the following steps were taken:

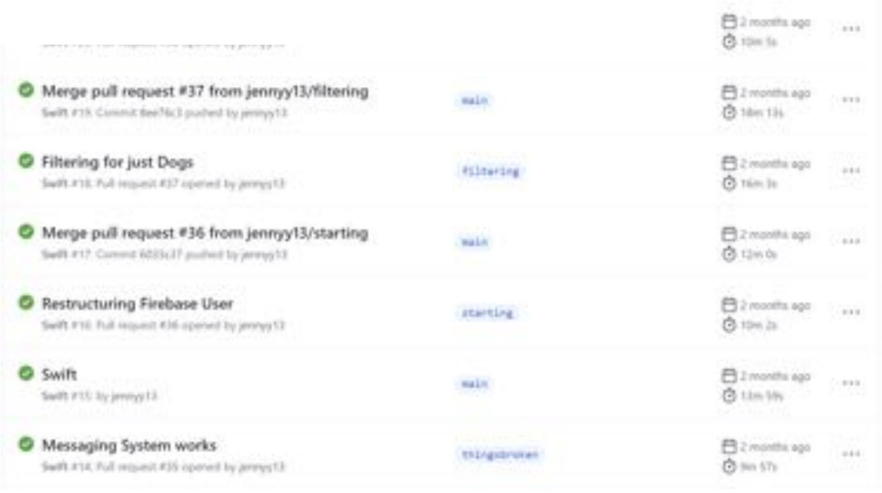
- Created a YAML file to define the environment for the test to run in, the test pre build steps and the test build steps.
- Configured the project with a unit test bundle and set up a new target.
- Added the workflow on GitHub actions.

While doing this, there were several errors occurring. It took several iterations to get the YAML file correct. Available simulators had to be searched through to find the correct one so it could be specified in the YAML file. Also, an extra step had to be added to download the device pre-set up.



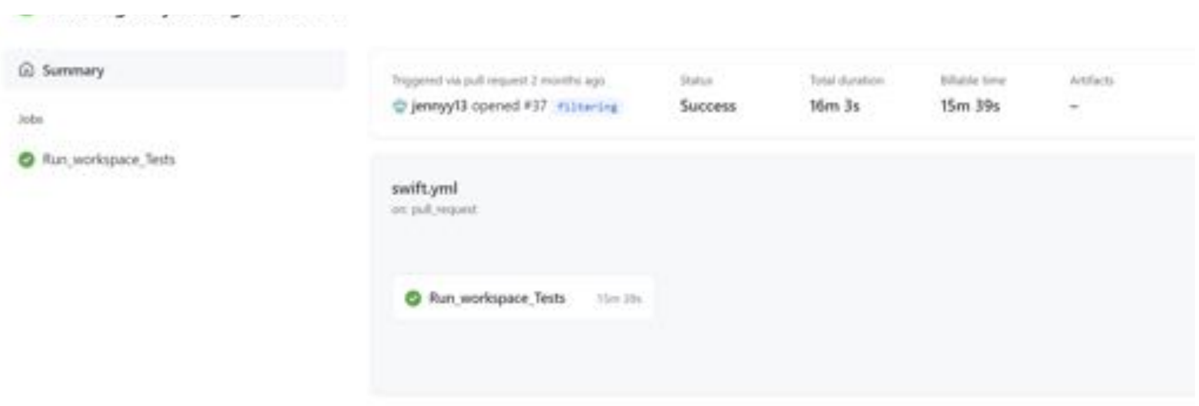
```yaml
name: Swift

on:

  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  Run_workspace_Tests:
    runs-on: macOS-latest
    steps:
    - name: pre-start simulator
      run: xcrun instruments -w "iPhone 11 (14.4) [23259447-2427-4881-8678-9C3030FF2821] (Simulator)" || true
    - uses: actions/checkout@v1
    - name: List available Xcode versions
      run: ls /Applications | grep Xcode
    - name: Select Xcode
      run: sudo xcode-select -switch /Applications/Xcode_12.1.app && /usr/bin/xcodebuild -version
    - name: Run unit tests
      run: xcodebuild test -scheme FYPJennifer -workspace FYPJennifer.xcworkspace -destination 'platform=iOS Simulator,name=iPhone 11,OS=14.4' | xcpretty && exit ${PIPESTATUS[
```

*Figure 51: Swift.yml file from project, for CI/CD*



| | | | |
|---|---|---|---|
| | | 2 months ago<br>10m 5s | ... |
| Merge pull request #37 from jennyy13/filtering<br>Swift #19: Commit 8ee76c3 pushed by jennyy13 | main | 2 months ago<br>18m 13s | ... |
| Filtering for just Dogs<br>Swift #18: Pull request #37 opened by jennyy13 | filtering | 2 months ago<br>16m 3s | ... |
| Merge pull request #36 from jennyy13/starting<br>Swift #17: Commit 6033c37 pushed by jennyy13 | main | 2 months ago<br>12m 0s | ... |
| Restructuring Firebase User<br>Swift #16: Pull request #36 opened by jennyy13 | starting | 2 months ago<br>10m 2s | ... |
| Swift<br>Swift #15: by jennyy13 | main | 2 months ago<br>13m 59s | ... |
| Messaging System works<br>Swift #14: Pull request #35 opened by jennyy13 | thingsbroken | 2 months ago<br>9m 57s | ... |

*Figure 52: Screenshot of Tests Passing on branches*

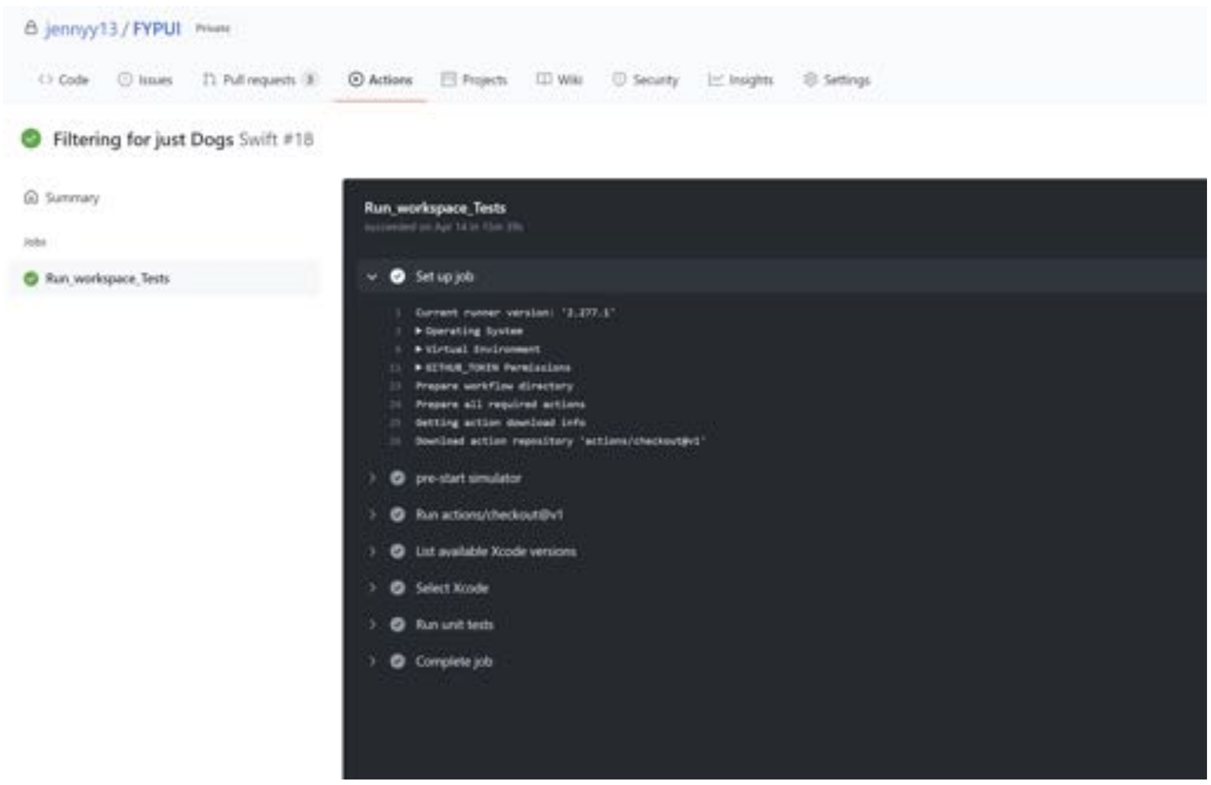


| Triggered via pull request 2 months ago | Status | Total duration | Billable time | Artifacts |
|---|---|---|---|---|
| jennyy13 opened #37 filtering | Success | 16m 3s | 15m 39s | – |

swift.yml
on: pull_request

Run_workspace_Tests 15m 39s

*Figure 53: Example of Job Workspace in GitHub actions*


jennyy13 / FYPUI Private

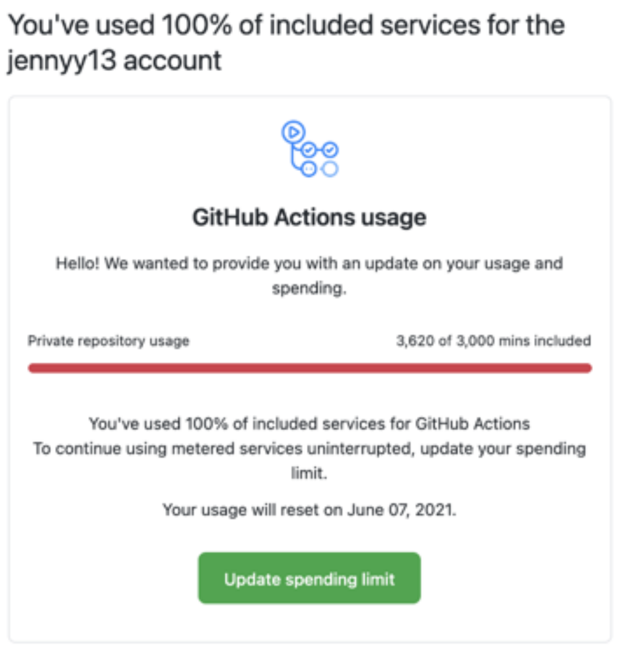
Filtering for just Dogs Swift #18

Run_workspace_Tests

```
Current runner version: '2.277.1'
▶ Operating System
▶ Virtual Environment
▶ GITHUB_TOKEN Permissions
Prepare workflow directory
Prepare all required actions
Getting action download info
Download action repository 'actions/checkout@v1'
```

*Figure 54: Workspace tests in GitHub actions that have passed*

However, unfortunately a number of large changes were made to the application, for example, the addition of new CocoaPods. This led to extremely long durations of these tests and this used up the private repository limit that GitHub sets. This means that the unit test no longer runs on new PRs .


You've used 100% of included services for the jennyy13 account

GitHub Actions usage

Hello! We wanted to provide you with an update on your usage and spending.

Private repository usage 3,620 of 3,000 mins included

You've used 100% of included services for GitHub Actions
To continue using metered services uninterrupted, update your spending limit.

Your usage will reset on June 07, 2021.

*Figure 55: Email about GitHub usage being exceeded*

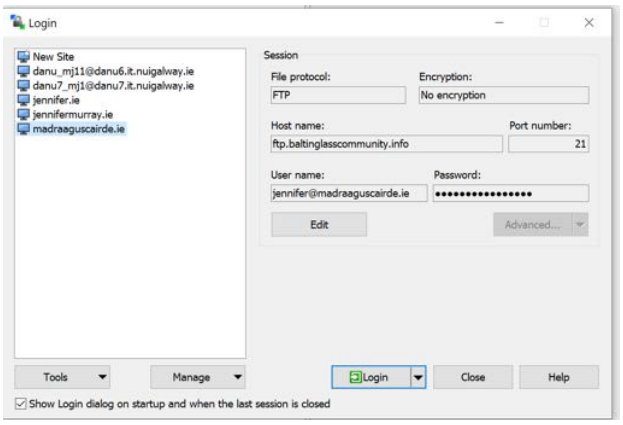
# Chapter 6. **Website**

## 6.1. **Introduction**

A website was created to accompany the app. The website does not have the same functionality as the app; however, it has all the set up done so that it could potentially be developed further in the future. For the scope of this project, it was kept simple but effective – to demonstrate a variety of skills acquired.

## 6.2. **Hosting**

To begin, the domain *madraaguscairde.ie* was registered through LetsHost, with its nameservers set to those of an existing hosting account. At the backend, the cPanel interface was used to add the domain to the hosting account, which set up the folders for the web server and added the domain to the Domain Name System (DNS). Via cPanel, a file transfer protocol (FTP) account was created. The FTP account credentials were added to WinSCP on the development computer to transfer files to the web server.

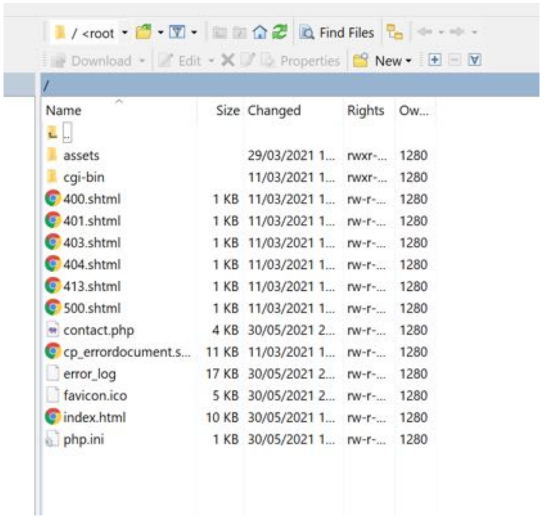


New Site
danu_mj11@danu6.it.nuigalway.ie
danu7_mj1@danu7.it.nuigalway.ie
jennifer.ie
jennifermurray.ie
madraaguscairde.ie

Session
File protocol: FTP
Encryption: No encryption
Host name: ftp.baltinglasscommunity.info
Port number: 21
User name: jennifer@madraaguscairde.ie
Password: ••••••••••••••••

☑ Show Login dialog on startup and when the last session is closed

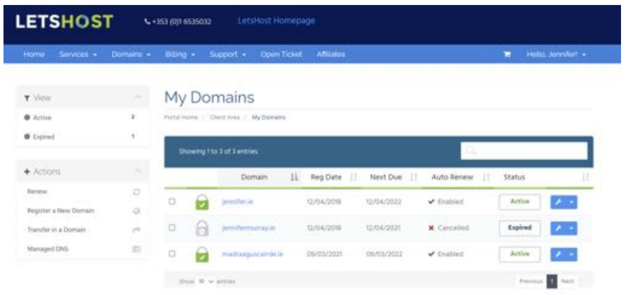
Figure 56: Login for WinSCP



| Name | Size | Changed | Rights | Ow... |
|---|---|---|---|---|
| .. | | | | |
| assets | | 29/03/2021 1... | rwxr-... | 1280 |
| cgi-bin | | 11/03/2021 1... | rwxr-... | 1280 |
| 400.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| 401.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| 403.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| 404.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| 413.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| 500.shtml | 1 KB | 11/03/2021 1... | rw-r-... | 1280 |
| contact.php | 4 KB | 30/05/2021 2... | rw-r-... | 1280 |
| cp_errordocument.s... | 11 KB | 11/03/2021 1... | rw-r-... | 1280 |
| error_log | 17 KB | 30/05/2021 2... | rw-r-... | 1280 |
| favicon.ico | 5 KB | 30/05/2021 2... | rw-r-... | 1280 |
| index.html | 10 KB | 30/05/2021 1... | rw-r-... | 1280 |
| php.ini | 1 KB | 30/05/2021 1... | rw-r-... | 1280 |

Figure 57: Files from Madra agus Cairde on WinSCP

The DNS configuration was checked by looking the name up on the specified name servers (NS5.DNSIRELAND.COM, NS6.DNSIRELAND.COM, NS7.DNSIRELAND.IE), then confirmed by entering the domain name into a web browser.



My Domains

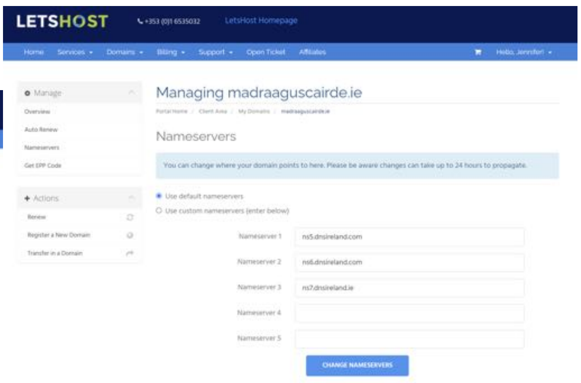
Portal Home / Client Area / My Domains

Active 2
Expired 1

Showing 1 to 3 of 3 entries

| Domain | Reg Date | Next Due | Auto Renew | Status |
|---|---|---|---|---|
| jennifer.ie | 12/04/2018 | 12/04/2022 | ✓ Enabled | Active |
| jennifermurray.ie | 12/04/2018 | 12/04/2021 | ✗ Cancelled | Expired |
| madraaguscairde.ie | 09/03/2021 | 09/03/2022 | ✓ Enabled | Active |

*Figure 58: Domain for Madra agus Cairde from Lets Host*



Managing madraaguscairde.ie

Portal Home / Client Area / My Domains / madraaguscairde.ie

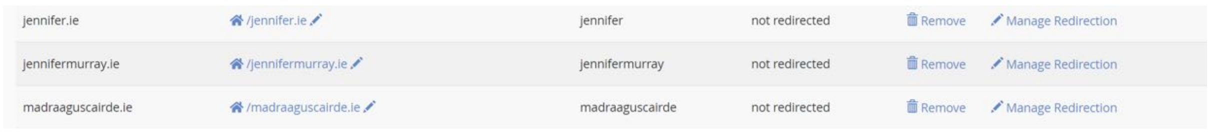
Nameservers

You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.

● Use default nameservers
○ Use custom nameservers (enter below)

| | |
|---|---|
| Nameserver 1 | ns5.dnsireland.com |
| Nameserver 2 | ns6.dnsireland.com |
| Nameserver 3 | ns7.dnsireland.ie |
| Nameserver 4 | |
| Nameserver 5 | |

*Figure 59: Namesevers for managing Madra agus Cairde*

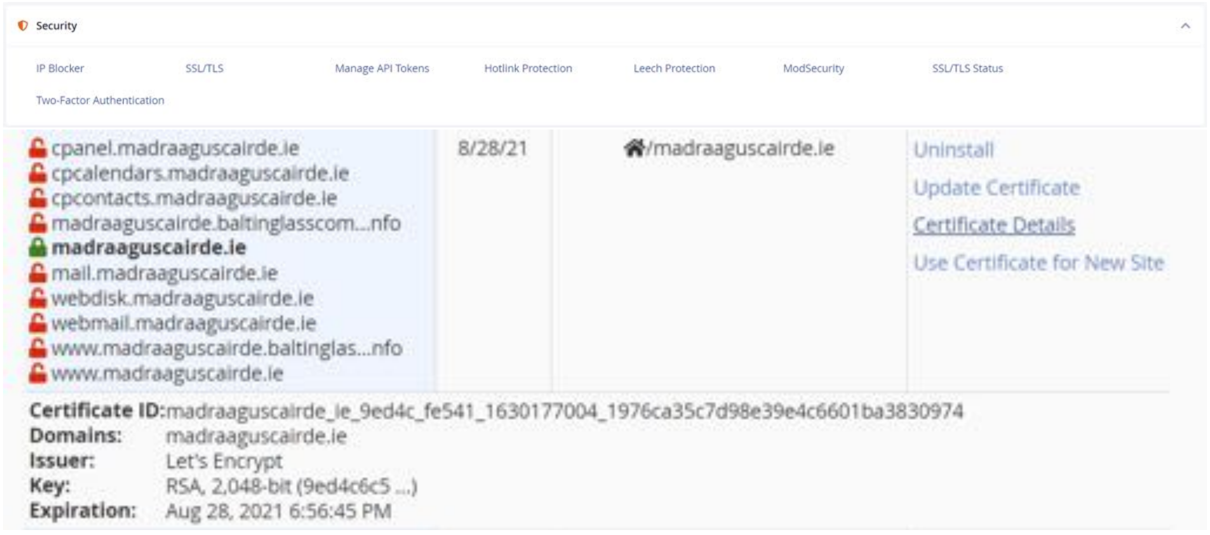


| | | | | | |
|---|---|---|---|---|---|
| jennifer.ie | /jennifer.ie | jennifer | not redirected | Remove | Manage Redirection |
| jennifermurray.ie | /jennifermurray.ie | jennifermurray | not redirected | Remove | Manage Redirection |
| madraaguscairde.ie | /madraaguscairde.ie | madraaguscairde | not redirected | Remove | Manage Redirection |

*Figure 60: Madra agus Cairde Domain*

## 6.3. SSL Certificate

When setting up *madraaguscairde.ie*, security was a key concern. To provide a trustworthy connection, an SSL certificate was created for the domain by Let's Encrypt. The SSL certificate ensures users can connect securely to visit the website. This means that no malicious website can fraudulently pretend to be *madraaguscairde.ie*. SSL certificates are a way of proving that the connection is to the right site. It works by getting the certificate from the website and ensuring that it matches the name of the requested website. SSL certificates also allow web traffic to be encrypted to avoid eavesdropping. Pages on the website contain scripts to redirect browsers to the secure (HTTPS) version if the plain (HTTP) version is requested.



cpanel.madraaguscairde.ie
cpcalendars.madraaguscairde.ie
cpcontacts.madraaguscairde.ie
madraaguscairde.baltinglasscom...nfo
**madraaguscairde.ie**
mail.madraaguscairde.ie
webdisk.madraaguscairde.ie
webmail.madraaguscairde.ie
www.madraaguscairde.baltinglas...nfo
www.madraaguscairde.ie

8/28/21

/madraaguscairde.ie

**Certificate ID:** madraaguscairde_ie_9ed4c_fe541_1630177004_1976ca35c7d98e39e4c6601ba3830974
**Domains:** madraaguscairde.ie
**Issuer:** Let's Encrypt
**Key:** RSA, 2,048-bit (9ed4c6c5 ...)
**Expiration:** Aug 28, 2021 6:56:45 PM

*Figure 61: Details of SSL certificates for Madra agus Cairde*

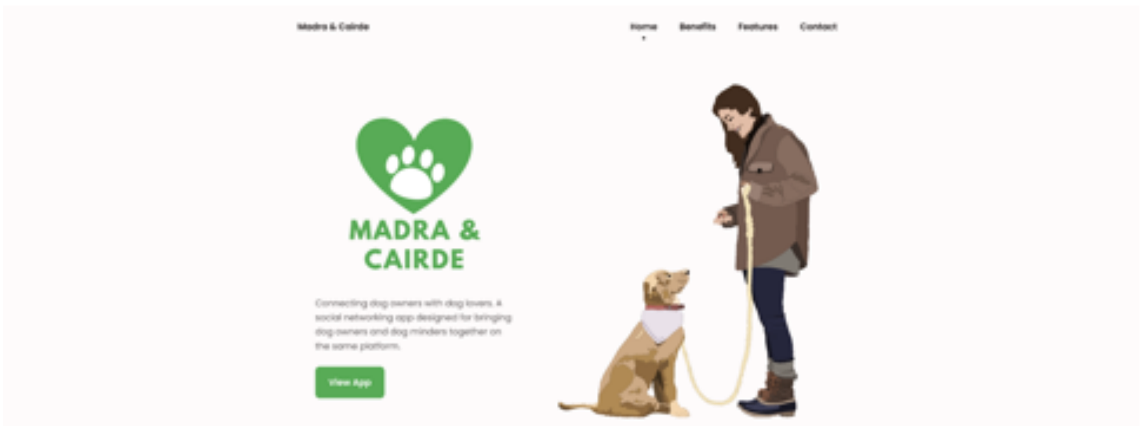
## 6.4. **User Interface**

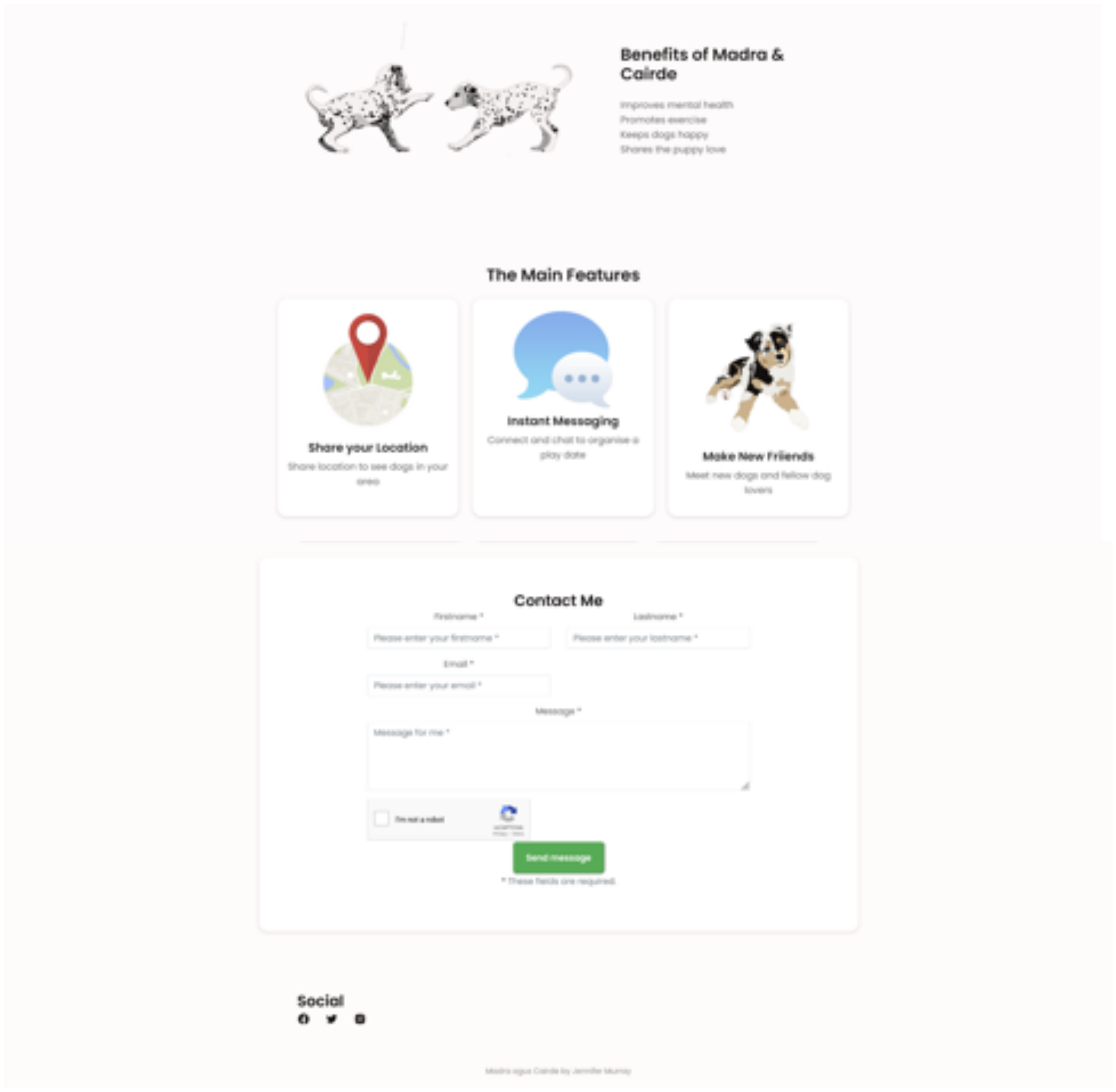
The importance of a good user interface (and UX) was described in the Design chapter (Chapter 3). Although the website does not have the same functionality of the app, it was felt necessary that it be as closely related to the app as possible, in terms of design. It needed to be obvious that they were aspect of the same brand, Madra & Cairde. This meant that when it came time to design the UI of the website, elements that were used to design the application's UI would similarly be implemented for the website.

The Madra agus Cairde logo is emphasised at the top of the website, which helps promote the brand design and bridge the gap between the website and the app. The same colour scheme was used which meant the website has the same advantages for user accessibility, particularly those with poor vision. The artwork that was drawn for the application also makes an appearance on the website, so ensure a cohesive and aesthetic appearance of the UI.

The UI of the website was based on an a YouTube tutorial [6] which demonstrated the creation of a website. This tutorial was closely followed as the look of the website was exactly what had been envisioned for the site. It was adapted to suit the specific needs that the Madra agus Cairde website needed. The website makes use of several useful web development libraries, including Bootstrap, which contribute greatly to the overall user experience.

To further adapt the website, to make it more unique, extra desirable features were added. The carousel on the website's second page was added to the website when the application development part of the project was complete. This seemed like a great way to make the website more unique and is a great way of demoing the app, seeing as it has not been published on the App store.


Madra & Cairde
Home
Benefits
Features
Contact
MADRA & CAIRDE
Connecting dog owners with dog lovers. A social networking app designed for bringing dog owners and dog minders together on the same platform.
View App



Benefits of Madra & Cairde

Improves mental health
Promotes exercise
Keeps dogs happy
Shares the puppy love

The Main Features

Share your Location
Share location to see dogs in your area

Instant Messaging
Connect and chat to organise a play date

Make New Friends
Meet new dogs and fellow dog lovers

Contact Me

| | |
|---|---|
| Firstname * | Lastname * |
| Please enter your firstname * | Please enter your lastname * |
| Email * | |
| Please enter your email * | |
| Message * | |
| Message for me * | |

I'm not a robot

Send message

* These fields are required

Social

Madra agus Cairde by Jennifer Murray

*Figure 62, 63, 64: Madra agus Cairde index.html*

In order to access the demo (or preview) of the app on the website, the 'View App' button should be clicked on the main screen. Screenshots from the app are organised into a carousel that can be scrolled through to view all the sample screens. This carousel was adapted from a tutorial found online [8].

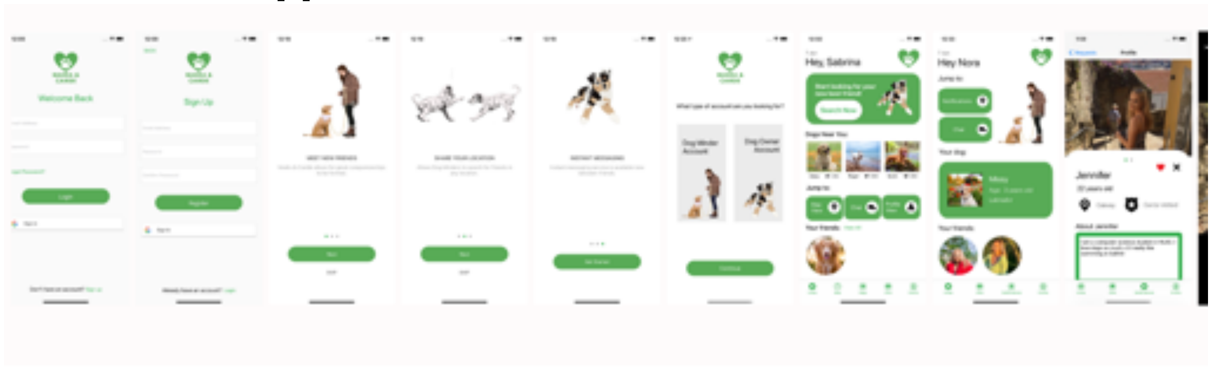


*Figure 65: Madra agus Cairde demo.html*

If the mouse hovers over an image, the image is enlarged and made to contrast with the other images, to highlight that specific one. The title of the image can be read; this describes the image briefly e.g., ‘Dog Minder Home Page’ so it is clear which page it represents in the app.

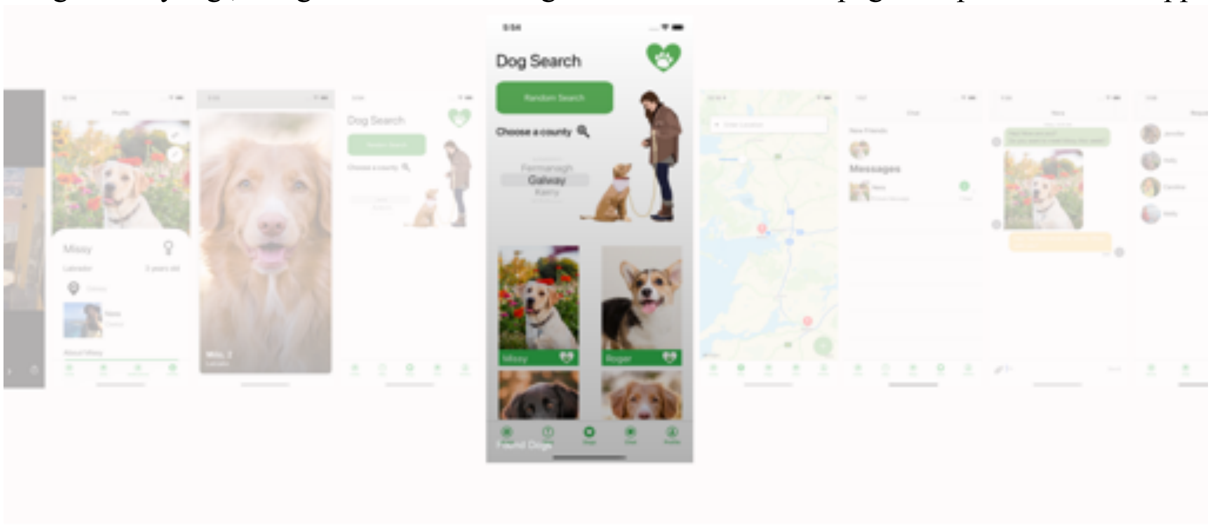


*Figure 66: Demo of the Carousel*

The favicon was kept in line with the signature Madra agus Cairde logo.

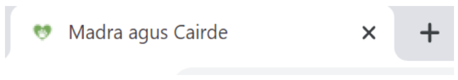


*Figure 67: Madra agus Cairde fav.ico*

### 6.5. **Contact Form**

As an extra feature, a support email account was set up for the domain (support@madraaguscairde.ie), linked to the author’s personal gmail account. With this support email address created, a contact form could be added to the website. In theory, it can be used by anyone who has questions about *Madra agus Cairde*. The contact form was a good way to add an interactive section to the website to demonstrate more web development skills. The contact form was based on an online tutorial [7]. An HTML contact form using the Bootstrap 4 framework uses AJAX to send contact details to a PHP script on the server. The benefit of using AJAX is that the whole page does not need to be reloaded when a person submits the contact form. The outline of the PHP file provided in the tutorial needed to be customised to work with the particular contact form created for the website. It needed to be hooked up to the support email and include the various form fields. There were several options for sending an email with PHP, such as PHPMailer, but the regular PHP mail() function worked adequately. After some trial and error, the contact form worked as desired.

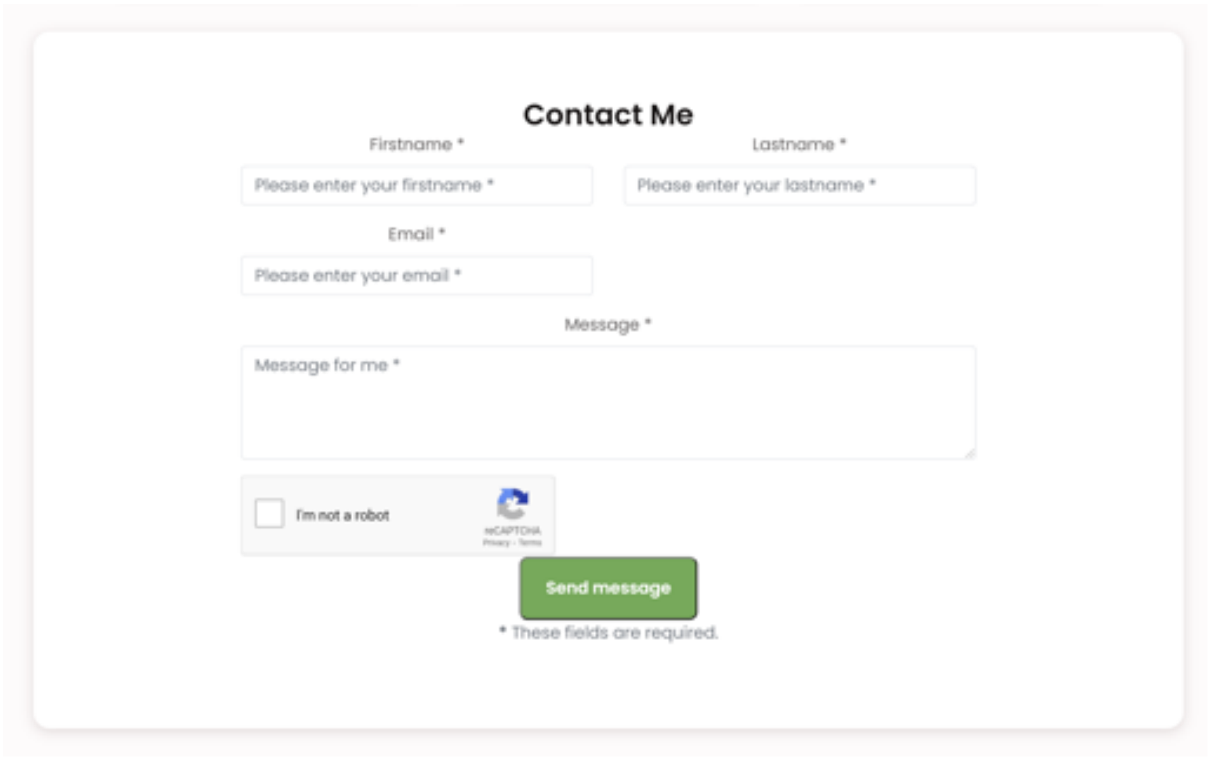
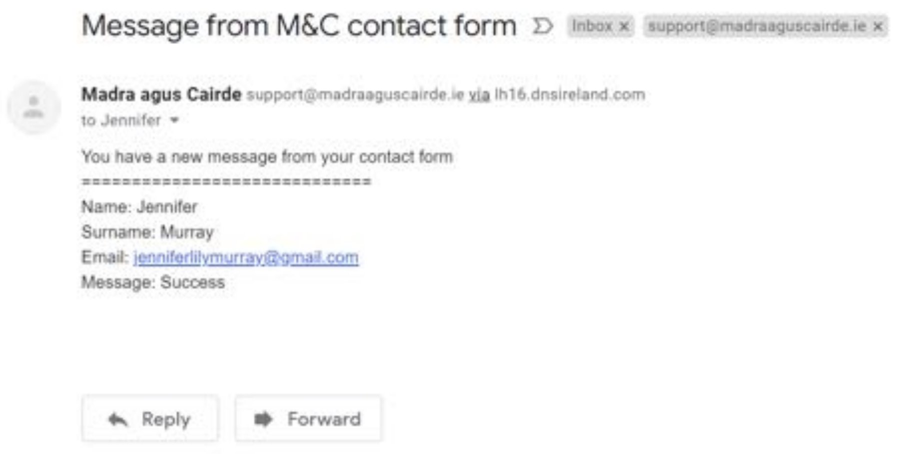


*Figure 68: Contact form on Madra agus Cairde website*



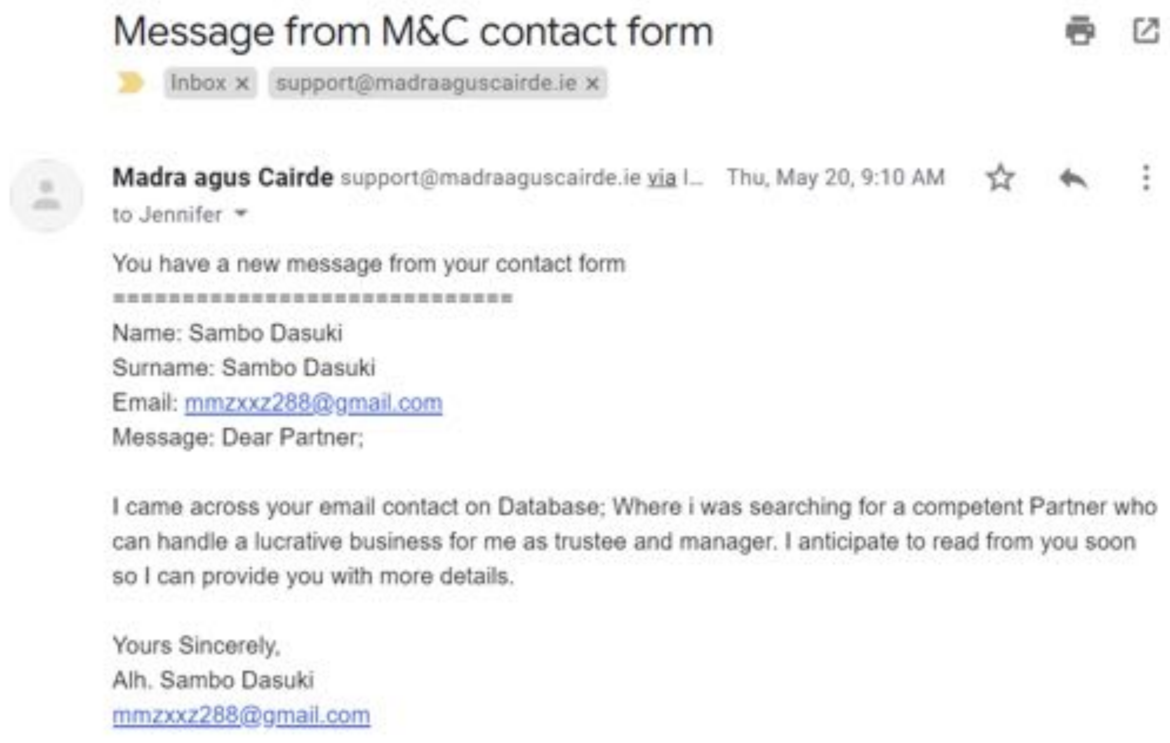
Madra agus Cairde support@madraaguscairde.ie via lh16.dnsireland.com
to Jennifer

You have a new message from your contact form
==============================
Name: Jennifer
Surname: Murray
Email: jenniferlilymurray@gmail.com
Message: Success

*Figure 69: Response from the contact form*

## 6.6. CAPTCHA

However, shortly after setting up the support email, it began to receive spam emails.



Message from M&C contact form

Inbox × support@madraaguscairde.ie ×

Madra agus Cairde support@madraaguscairde.ie via l... Thu, May 20, 9:10 AM

to Jennifer

You have a new message from your contact form

```
==============================
```

Name: Sambo Dasuki

Surname: Sambo Dasuki

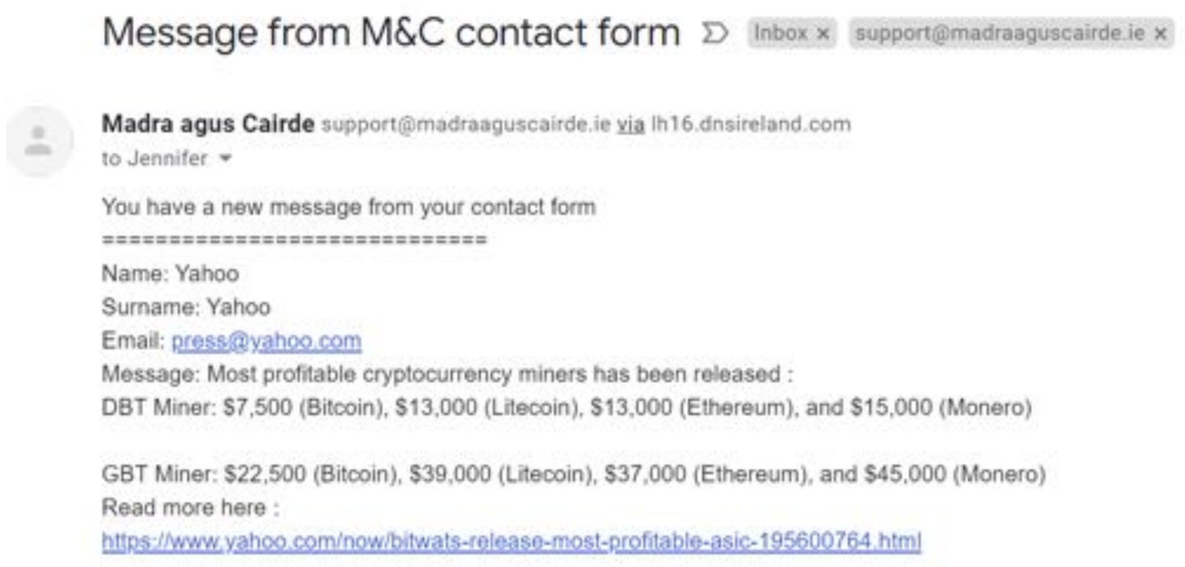
Email: mmzxxz288@gmail.com

Message: Dear Partner;

I came across your email contact on Database; Where i was searching for a competent Partner who can handle a lucrative business for me as trustee and manager. I anticipate to read from you soon so I can provide you with more details.

Yours Sincerely,
Alh. Sambo Dasuki
mmzxxz288@gmail.com

*Figure 70: Spam email example*



Message from M&C contact form Inbox × support@madraaguscairde.ie ×

Madra agus Cairde support@madraaguscairde.ie via lh16.dnsireland.com

to Jennifer

You have a new message from your contact form

```
==============================
```

Name: Yahoo

Surname: Yahoo

Email: press@yahoo.com

Message: Most profitable cryptocurrency miners has been released :
DBT Miner: $7,500 (Bitcoin), $13,000 (Litecoin), $13,000 (Ethereum), and $15,000 (Monero)

GBT Miner: $22,500 (Bitcoin), $39,000 (Litecoin), $37,000 (Ethereum), and $45,000 (Monero)
Read more here :
https://www.yahoo.com/now/bitwats-release-most-profitable-asic-195600764.html

*Figure 71: Spam email example*

The emails kept arriving and proved an annoyance and distraction. There are spam bots which crawl the web to find contact forms like the one on this website and fill them with spam. The most common type of spam received via this address related to crypto-currencies. After researching possible ways to deal with spam, the most promising approach appeared to be to add a CAPTCHA to the contact form.

CAPTCHA stands for 'Completely Automated Public Turing test to tell Computers and Humans Apart'. It is basically an extra form of security and verification to ensure that a visitor to a website is probably not a robot.

Adding this to the website was done by following example code [9] but a few problems were encountered as some things did not work as expected.

Google reCAPTCHA has two ways to use it; v2 and v3, with v2 being considered the most suitable for this project. Google reCAPTCHA v3 does not actually challenge people on whether they are a robot, but just assesses various parameters, such as movement on the website, to give a score based on how likely the user is to be a human. Having a more explicit CAPTCHA was felt to be a better deterrent to bots filling in the form. For human users, a simple checkbox is normally all that is offered, but if there is doubt, it will offer a picture-challenge, as seen below.

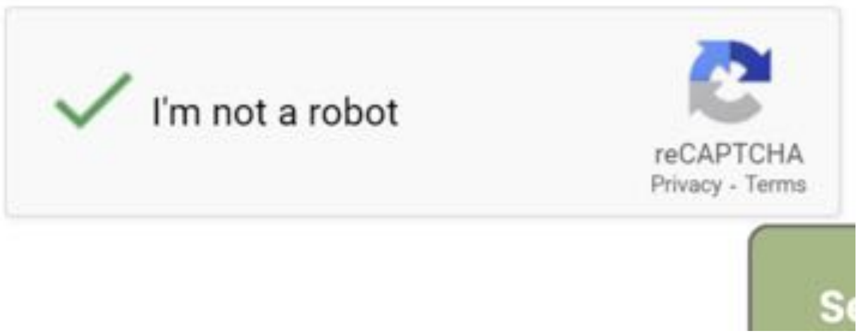


*Figure 72: Google reCAPTCHA implemented*

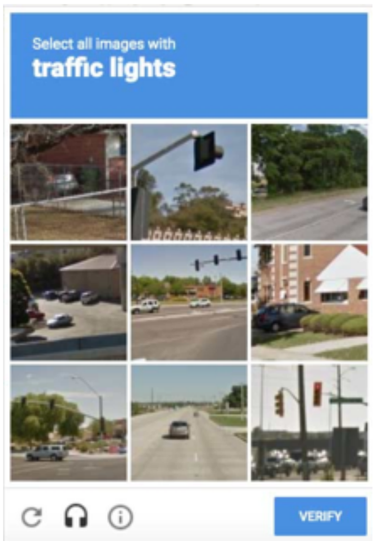


*Figure 73: reCAPTCHA picture test*

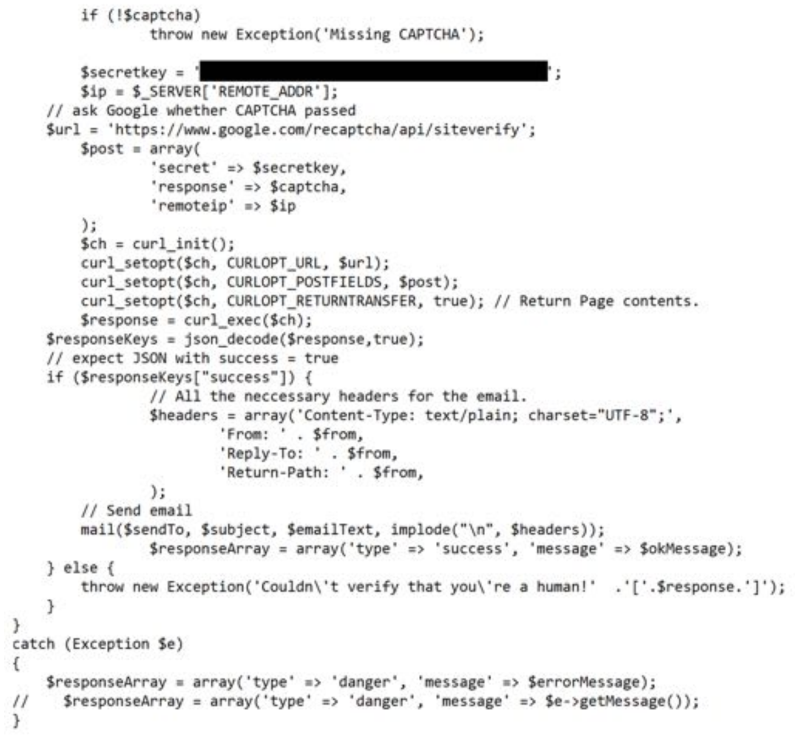
To use the Google reCAPTCHA service, a Google Developer account had to be created. Then a pair of keys (one public, one secret) could be created for *madraaguscairde.ie*. Adding the CAPTCHA to the HTML contact form was done simply by including a Google JavaScript file and adding a <div> element with a particular class name and the public key.

```
<div class="g-recaptcha col-md-6" data-
sitekey="6LfqPv8aAAAAAChsjFLz7b3hiCdOpxqhT5FSpYSe"></div>
```

This was enough on the client side to get it to appear, and the rest of the work was done server side. While the public key can be accessed by anyone looking at the page source, the secret key is only used in the server code, to verify the CAPTCHA's authenticity with Google.

A few problems were found when trying to implement the CAPTCHA. First, the CAPTCHA was never being verified. This turned out to be because the PHP function 'file_get_contents' was being used to get the JSON document from Google containing CAPTCHA status. After some investigation, it became apparent that by default that function does not work on URLs, only local files. However, the Google documentation says that calls to the reCAPTCHA API need to use the POST method, not GET which is what 'file_get_contents' uses. The problem was resolved by using 'cURL' instead.

Now, the contact form requires the user to satisfy Google that it is not a robot, otherwise the form will not send an email.

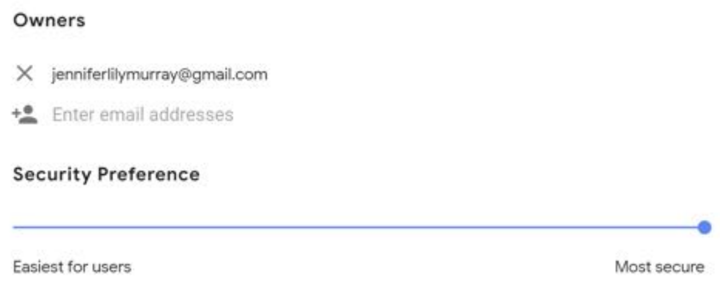


```
        if (!$captcha)
                throw new Exception('Missing CAPTCHA');

        $secretkey = '[redacted]';
        $ip = $_SERVER['REMOTE_ADDR'];
    // ask Google whether CAPTCHA passed
    $url = 'https://www.google.com/recaptcha/api/siteverify';
        $post = array(
                'secret' => $secretkey,
                'response' => $captcha,
                'remoteip' => $ip
        );
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true); // Return Page contents.
        $response = curl_exec($ch);
    $responseKeys = json_decode($response,true);
    // expect JSON with success = true
    if ($responseKeys["success"]) {
                // All the neccessary headers for the email.
                $headers = array('Content-Type: text/plain; charset="UTF-8";',
                        'From: ' . $from,
                        'Reply-To: ' . $from,
                        'Return-Path: ' . $from,
                );
        // Send email
        mail($sendTo, $subject, $emailText, implode("\n", $headers));
                $responseArray = array('type' => 'success', 'message' => $okMessage);
    } else {
        throw new Exception('Couldn\'t verify that you\'re a human!'  .'['.$response.']');
    }
}
catch (Exception $e)
{
    $responseArray = array('type' => 'danger', 'message' => $errorMessage);
//    $responseArray = array('type' => 'danger', 'message' => $e->getMessage());
}
```

*Figure 74: PHP code for sending emails after CAPTCHA verification*

*(Secret key is blacked out as a safety measure – as learned in Computer Security module)*

On the Google Developer control panel for the reCAPTCHA settings, there is a slider to adjust the security preference between least intrusive and most secure. This site now uses the most secure one because it results in the fewest spam emails.


Owners

jenniferlilymurray@gmail.com

Security Preference

*Figure 75: Security preference setting for reCAPTCHA*

# Chapter 7. **Conclusion**

## 7.1. **Challenges**

This project was a very big undertaking and it did present a number of challenges along the way, some of which were more complicated than others. A number of solutions were implemented to combat several of these challenges. However, other challenges were too big to tackle in the available time frame, and these will be detailed below. Overall, a lesson was learned from each of the challenges and a lot could be avoided in future if a similar project were to be undertaken.

### COVID-19

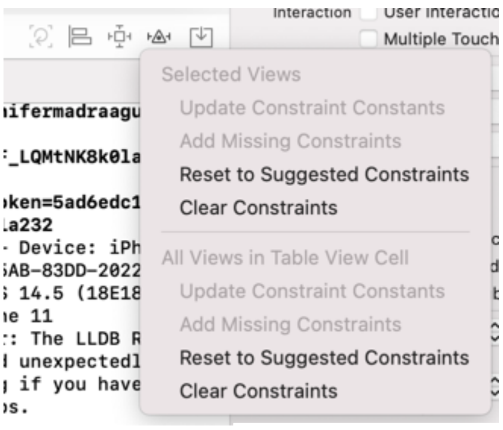
As with everyone's Final Year Project, COVID-19 did present as an enormous challenge during the development of this project. Firstly, the lack of study space in college made it quite challenging to be able to do FYP work. It took several weeks to organise a proper desk and work-from-home set up, which definitely hindered progress on FYP from the beginning. It also meant that the in-person support like DISC was not as accessible as it would have been had students been on campus. Lastly, the frequent lengthy lock-downs meant it was hard to ask friends or family for feedback. Luckily, housemates were more than willing to be user testers and Zoom became useful for getting feedback from other friends.

### Testing

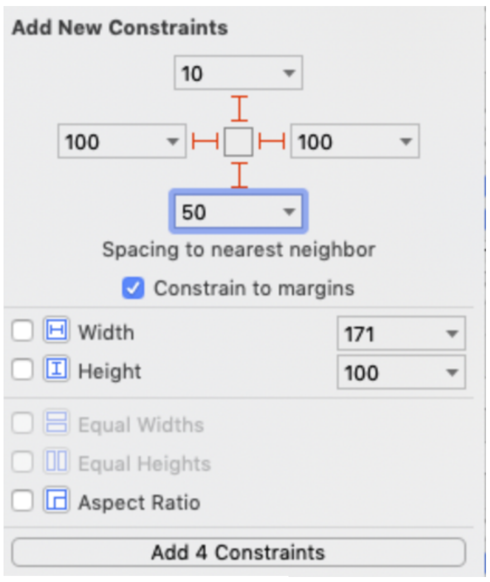
Due to COVID-19 and the continuous lockdowns over the semester, it was quite challenging to arrange thorough user testing. A lot of acceptance testing therefore had to be run by the author, to ensure any logical changes to the code worked as expected. Where possible, Zoom meetings were hosted and user testing was done this way. It was not ideal, but the only way around it in the midst of a pandemic.

### New Environment

It is difficult to pinpoint a principal reason leading to insufficient time to complete all of the desired features. Perhaps it was overly ambitious to plan for so many features in a project while working in a completely new environment. There was a very difficult learning curve when it came to developing in Xcode. A lot of things that are almost second nature now were completely new at first and took months to comprehend. One thing that has still not gotten any easier with practice is using the auto-layout constraints for view controllers. It makes it very difficult to set items in the precise location the item is needed.


Interaction
User Interactio
Multiple Touch
Selected Views
Update Constraint Constants
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
All Views in Table View Cell
Update Constraint Constants
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints

```
ıifermadraagu
:_LQMtNK8k0la
ıken=5ad6edc1
ıa232
- Device: iPh
;AB-83DD-2022
; 14.5 (18E18
ıe 11
:: The LLDB R
ı unexpectedl
ı if you have
ıs.
```



Top: 10
Left: 100
Right: 100
Bottom: 50
[✓] Constrain to margins
[ ] Width 171
[ ] Height 100
[ ] Equal Widths
[ ] Equal Heights
[ ] Aspect Ratio

*Figure 76, 77: Auto Layout Constraints in Xcode*

Database:

Originally, when researching what databases to use, it seemed to be suggested that Cloud Firestore was the better alternative to the Realtime Database. However, it was discovered too late that the map feature that was planned to be implemented, needed the Realtime Database to work. GeoFire is a library that allows the storage and querying of keys based on geographic location. This would have been exactly what was needed to implement the map feature which would work by showing the Dog Minder dogs in their area. GeoFire uses Realtime Database not Cloud Firestore, so it was not possible to do this feature. A prototype version of this was made for the app for demo purposes. For this, a set of 'dummy' dogs were created and their locations represented by pinpoints on the map.

Technical Difficulties:

Unfortunately, technical difficulties accounted for a large number of the challenges faced during this project. Xcode is very demanding in terms of computer resources and energy and would often crash out and be unusable for hours. iOS development with Xcode and Swift requires use of an Apple Mac. The MacBook Pro available was not a very advanced model as it was purchased on student resources with just one year of college left in mind. Therefore, the more powerful (and expensive) versions were not an option. This was by far the greatest impediment to development as the only way to resolve the issue was to switch the laptop off for hours, which clearly hindered work on this project, and other college work which needed to be done.

Below are just two (of several) types of errors that would often occur.

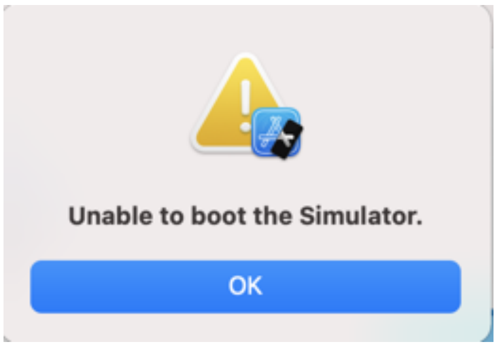
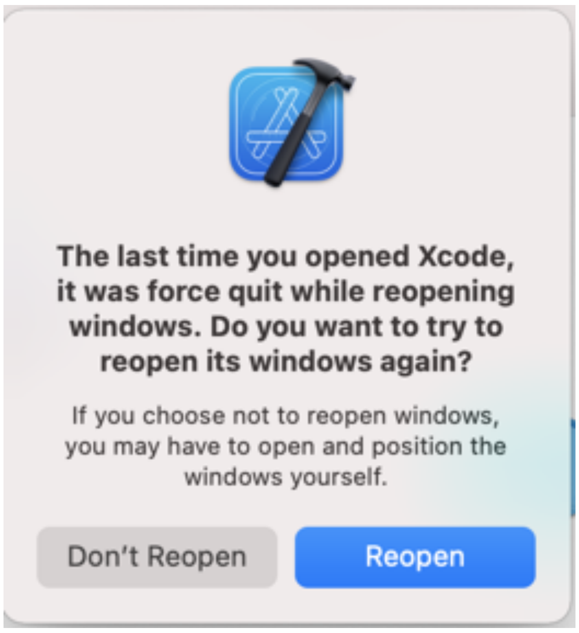

Unable to boot the Simulator.

*Figure 78: Warning that simulator was unable to boot*


The last time you opened Xcode, it was force quit while reopening windows. Do you want to try to reopen its windows again?

If you choose not to reopen windows, you may have to open and position the windows yourself.

*Figure 79: Warning of a Force Quit in Xcode*

Often, if an error appeared and seemed untraceable after investigation, the only way to solve it was to delete the 'derived data' folder. A clean build was needed after this was done and could take several hours, depending on how slow the computer was working at the time. This proved a great nuisance, not to mention the days wasted when first trying to track down the cause of the error.

These technical difficulties were unavoidable with the computer used so no real lesson could have been learned from this challenge.

## 7.2. **Lessons Learned**

This project has been a great learning experience. The lessons learned have inspired confidence in working on any similar future project. Even if it is not in the same realm as the scope of this project, many lessons learned are universally applicable, such as user testing and project management. Also, having to learn new tools from scratch will be less daunting after doing so much of that here.

### Technical Skills

iOS mobile development was a completely new territory when this project began. Technical skills and knowledge have come a long way. Competency in Swift has been gained as well as familiarity and comfort with XCode. Also, debugging skills have majorly improved as they proved necessary for fixing bugs experienced during development.

The skills needed for publishing and securing a website were also acquired. Bootstrap framework was made use of to develop a responsive webpage. A good understanding of AJAX was gained for adding interactive elements to the web page. PHP was learned and used for server side scripts.

Overall, a deeper knowledge of the inner workings of mobile applications and how to develop them, as well as what it takes to build and publish a website was gained.

User Testing

Throughout the course of this project, many user tests were held. These proved invaluable in improving user experience. It was necessary to develop the ability to interpret results from these tests and translate them into improvements to the design and code. User testing provided feedback which led to a greater understanding of what makes a good user experience.

Project Management

The importance of good project management was made clear by this project. This project highlighted the importance of sticking to good project management strategies and demonstrated how they benefit projects as a whole. Project management demonstrated that the scope of the project needed to be limited in order to be able to achieve the deadlines. It still ensured a viable project was left at the end because proper prioritisation of tasks was done. Using project management tools such as Trello became second nature after growing familiar with them and they became essential elements of the project.

Design Prototyping

Useful tools such as Adobe XD and Miro were used for UX research. The need for good design prototyping became apparent early on as it was necessary to have a good idea of what the screens would look like when developing the basis for the app. Prototyping helps ensure that the functional requirements of the app will be met. It is also a great way to trial the positions of buttons and colour schemes until a pleasing and functional design is found. It is much easier and less time consuming to test these ideas out on a program used for prototyping compared to building one from scratch each time in Xcode.

## 7.3. Future Work

Madra agus Cairde is an exciting new idea which could lead to an extremely viable product, if it were to go to market. Through the designing and implementing of Madra agus Cairde, a lot of lessons have been learned and these have fuelled the way forward. There are a number of different avenues which would be good to expand on which will be outlined next.

**Android:**

Originally the iOS platform was chosen for Madra agus Cairde. However, the next logical step, if Madra agus Cairde was going to expand, would be to develop it out to be used on Android devices as well.

**Website:**

A lot of the groundwork for the website has already been done, including the hosting and the SSL certificates. As of now, it is just a simple set up with information about the app and a contact form. Eventually, the plan would be to add the same functionality from the app onto the website. This would make Madra agus Cairde more universal and accessible, as older generations tend to find websites easier to use than apps. It would expand the reach of Madra agus Cairde.

**Unit Tests:**

A unit test is already set up to ensure failing code cannot be merged onto the main branch. However, when there is time in the future, to ensure the integrity of the code more tests would need to be implemented to test specific functionality throughout the app.

**GeoFire (Realtime Database):**

As mentioned in the Challenges section, GeoFire would be an ideal way of implementing the Map Feature correctly. If the database that is supporting the app could be switched over to a Realtime Database in future, then the GeoFire library could be utilised and the feature implemented correctly.

**International:**

The target audience for this app was Ireland. The app only allows for counties on the island of Ireland to be set as the location. This would be an area that would be very beneficial to extend. If the app was scaled in this way it would allow the app to be used internationally.

**Social Logins:**

Social login is a software design pattern that enables users to authenticate themselves on different applications and sites by linking through a social networking site rather than inputting a separate ID and password for each application/website.
Having a social login extension on the Madra & Cairde App/ website would benefit the app greatly. Along with enhancing the user's experience on the site, it would also encourage users to engage more. Another benefit to the social login extension being included on Madra & Cairde is that it would allow marketers to gather more accurate information, including age, relationship status and interests.
An attempt at implementing a Google login to app was made, although the attempt was unsuccessful, and time did not allow for further attempts.

**Review Section:**

Another feature that should be considered in future for the app is a review section. Having originally planned for it to be part of the app, it unfortunately had to be cut from the feature list due to lack of time availability.
Online reviews offer many benefits, including free advertising, improved search engine results and constructive criticism and suggestions.

## 7.4. Final Conclusion

This report aimed to outline the project and describe how it was planned and executed. It provides a summary of what was achieved in the time available and details of the main fruits of the effort.

Valuable insights into the software development process were learned. In managing the process, the need for constantly revising achievable goals was realised.

The result is an app and basic website with the potential to be developed further into a real product. Developing it was time well spent, despite the occasional frustration and disappointment.

## Appendix

**GitHub Repository:**
- Madra agus Cairde App: https://github.com/jennyy13/FYPUI

- Madra agus Cairde Website: https://github.com/jennyy13/WebsiteMAC/settings

*Security key was accidentally uploaded to GitHub so these repositories cannot be made public as it can still be seen as deleted.*

*File have been uploaded to a Google Drive instead*

**Google Drive Link:**

https://drive.google.com/drive/folders/1gSPYOqPdaV0hc2dxRmQ-eMB0jqpwXyA0?usp=sharing

**Trello:**
- FYP Board:
https://trello.com/invite/b/QEMReL2C/04713d7b8b77eaa4770276396b19ec8a/fyp-to-do

**Website:**
- https://madraaguscairde.ie/

## References

[1] Dorfman MSW, Ph.D., D., 2020. The Health Benefits Of Pet Love. [online] Psychology Today. Available at: <The Health Benefits of Pet Love> [Accessed 29 November 2020].

[2]Sharma, Ashish et al, 2006. Exercise for Mental Health. [online] The Primary Care Companion to the
Journal of Clinical Psychiatry. Available at: <Exercise Improves Mental Health> [Accessed 29 November 2020]

[3] Bailey, L., 2020. Coping With Separation Anxiety In Dogs During COVID-19. [online] Psychology Today. Available at: <Coping With Separation Anxiety in Dogs During COVID-19> [Accessed 29 November 2020].

[4] Run Those Dogs. 2020. How Exercise Improves Pet Behavior - Run Those Dogs. [online] Available at: <How Exercise Improves Pet Behavior> [Accessed 29 November 2020].

[5] Tilda.tcd.ie. 2020. [online] Available at: <Report on Pet Ownership>[Accessed 29 November 2020].

**Additional Resources**

*Website Resources:*

[6] Foundations of the Website:
https://www.youtube.com/watch?v=RTIueV7zERY&t=2786s

[7] Contact Form: https://bootstrapious.com/p/how-to-build-a-working-bootstrap-contact-form

[8] Carousel: https://codepen.io/joshhunt/pen/LVQZRa

[9] reCAPTCHA tutorial: https://codeforgeek.com/google-recaptcha-tutorial/

Google reCAPTCHA: https://developers.google.com/recaptcha

*Application Resources:*

A lot of the learning and basis for the app was learned from the following videos:

- https://www.udemy.com/course/ios14-tinder-like-dating-application-with-firebase-swift/

- https://www.udemy.com/course/programmatic-uber-clone-swift-firebase-no-storyboards/

- https://www.udemy.com/course/build-full-realtime-chat-tinder-app/

- https://makeappicon.com/

****Code was adapted to suit the needs of the app. Proper referencing of work is done in the source code.****

*CocoaPod Libraries:*

SKPhotoBrowse: https://github.com/suzuki-0000/SKPhotoBrowser
Firebase: https://firebase.google.com/docs/ios/setup
Gallery: https://cocoapods.org/pods/Gallery
NVActivityIndicatorView/AppExtension:
https://cocoapods.org/pods/NVActivityIndicatorView
ProgressHUD: https://cocoapods.org/pods/ProgressHUD
Shuffle-iOS: https://cocoapods.org/pods/Shuffle-iOS